VISIT…

Roozbeh Hazrat

SPRINGER UNDERGRADUATE
MATHEMATICS SERIES

Mathematica®: A Problem-Centered Approach

S
U
M
S

Springer

Springer Undergraduate Mathematics Series

Advisory Board

M.A.J. Chaplain *University of Dundee*
K. Erdmann *University of Oxford*
A. MacIntyre *Queen Mary, University of London*
E. Süli *University of Oxford*
J.F. Toland *University of Bath*

For other titles published in this series, go to
www.springer.com/series/3423

Roozbeh Hazrat

Mathematica®: A Problem-Centered Approach

Springer

for Wolfram *Mathematica*

Roozbeh Hazrat
Department of Pure Mathematics
Queen's University
Belfast BT7 1NN
United Kingdom
r.hazrat@qub.ac.uk

ISSN 1615-2085
ISBN 978-1-84996-250-6 e-ISBN 978-1-84996-251-3
DOI 10.1007/978-1-84996-251-3
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010929694

Mathematics Subject Classification (2000): 68-01, 68N15

© Springer-Verlag London Limited 2010
Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licenses issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.
The use of registered names, trademarks, etc., in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant laws and regulations and therefore free for general use.
The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Mathematica and the *Mathematica* logo are registered trademarks of Wolfram Research, Inc ("WRI" – www.wolfram.com) and are used herein with WRI's permission. WRI did not participate in the creation of this work beyond the inclusion of the accompanying software, and it offers no endorsement beyond the inclusion of the accompanying software

Cover design: deblik

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

به پدرم و به مادرم

و به شهره و آزاده

Preface

> Teaching the mechanical performance of routine mathematical operations and nothing else is well under the level of the cookbook because kitchen recipes do leave something to the imagination and judgment of the cook but the mathematical recipes do not.
>
> G. Pólya

This book grew out of a course I gave at Queen's University Belfast during the period of 2004 to 2009. Although there are many books already written about how to use Wolfram *Mathematica*®, I noticed they fall into two categories: either they provide an explanation about the commands, in the style of: enter the command, push the button and see the result; or they study some problems and write several-paragraph codes in *Mathematica*. The books in the first category did not inspire me (or my imagination) and the second category were too difficult to understand and not suitable for learning (or teaching) *Mathematica*'s abilities to do programming and solve problems.

I could not find a book that I could follow to teach this module. In class one cannot go on forever showing students just how commands in *Mathematica* work; on the other hand it would be very difficult to follow the codes if one writes a program having more than five lines (especially as *Mathematica*'s style of programming provides a condensed code). Thus this book.

This book promotes *Mathematica*'s style of programming. I tried to show when we adopt this approach, how naturally one can solve (nice) problems with (*Mathematica*) style.

Here is an example: Does the Euler formula $n^2 + n + 41$ produce prime numbers for $n = 1$ to 39?

```
(#^2 + # + 41) & /@ Range[39] ∈ Primes
```

```
True
```

Or in another problem we tried to show how one can effectively use pattern

matching to check that for two matrices A and B, $(ABA^{-1})^5 = AB^5A^{-1}$. One only needs to introduce the fact that $AA^{-1} = 1$ and then *Mathematica* will check the problem by cancelling the inverse elements instead of direct calculation.

Although the meaning of the code above may not be clear yet, the reader will observe as we proceed how the codes start making sense, as if this is the most natural way to approach the problems. (People who approach problems with a procedural style of programming (such as C++) will experience that this style replaces their way of thinking!) We have tried to let the reader learn from the codes and avoid long and exhausting explanations, as the codes will speak for themselves. Also we have tried to show that in *Mathematica* (as in the real world) there are many ways to approach a problem and solve it. We have tried to inspire the imagination!

Someone once rightly said that the *Mathematica* programming language is rather like a "Swiss army knife" containing a vast array of features. *Mathematica* provides us with powerful mathematical functions. Along with this, one can freely mix different styles of programming, functional, list-based and procedural, to achieve a lot. This mélange of programming styles is what we promote in this note.

Thus this book could be considered for a course in *Mathematica*, or for self study. It mainly concentrates on programming and problem solving in *Mathematica*. I have mostly chosen problems having something to do with numbers as they do not need any particular background. Many of these problems were taken from or inspired by those collected in [3].

I would like to thank Ilan Vardi for answering my emails and Brian McMaster and Judith Millar for polishing the English of this note.

Naoko Morita encouraged me to make my notes into this book. I thank her for this and for always smiling and having a little Geschichte zu erzählen.

Roozbeh Hazrat
r.hazrat@qub.ac.uk
Belfast, October 2009

How to use this book

Each chapter of the book starts with a description of a new topic and some basic examples. We will then demonstrate the use of new commands in several problems and their solutions. We have tried to let the reader learn from the codes and avoided long and exhausting explanations, as the codes will speak for themselves.

There are three different categories of problems, shown by different frames:

Problem 0.1

These problems are the essential parts of the text where new commands are introduced to solve the problem and where it is demonstrated how the commands are used in Wolfram *Mathematica*®. These problems should not be skipped.

$\Longrightarrow$ Solution.

Problem 0.2

These problems further demonstrate how one can use commands already introduced to tackle different situations. The readers are encouraged to try out these problems by themselves first and then look at the solution.

$\Longrightarrow$ Solution.

Problem 0.3

These are more challenging problems that could be skipped in the first reading.

$\Longrightarrow$ Solution.

Most commands in *Mathematica* have several extensions. When we introduce a command, this is just the tip of the iceberg. In TIPS, we give further indications about the commands, or new commands to explore. Once one has enough competence, then it is quite easy to learn further abilities of a command by using the *Mathematica* Help and the examples available there.[1]

[1] Included with this book is a free 30 day trial of the *Mathematica* software. To access your free download, simply go to http://www.wolfram.com/books/resources and enter license number L3280-8445. You will be guided to download and install the latest version of *Mathematica*.

The *Mathematica* philosophy

In the beginning is the expression. Wolfram *Mathematica*® transforms the expression dictated by the rules to another expression. And that's how a new idea comes into the world!

The rules that will be used frequently can be given a name (we call them functions)

```
r[x_]:=1+x^2

r[arrow]
1+arrow^2

r[5]
26
```

And the transformation could take place immediately or with a delay

```
{x,x}/.x->Random[]
{0.0474307, 0.0474307}

{x,x}/.x:>Random[]
{0.368461, 0.588353}
```

The most powerful transformations are those which look for a certain pattern in an expression and morph that to a new pattern.

```
(a + b)^c /. (x_ + y_)^z_ -> (x^z + y^z)
a^c + b^c
```

And one of the most important expressions that we will work with is a list. As the name implies this is just a collection of elements (collection of other expressions). We then apply transformations to each element of the list:

```
x^ {1, 5, 7}
{x, x^5, x^7}
```

Any expression is considered as having a "head" followed by several arguments, `head[arg1,arg2,...]`. And one of the transformations which provide a capable tool is to replace the head of an expression by another head!

```
Plus @@ {a,b,c}
a+b+c

Power @@ (x+y)
x^y
```

Putting these concepts together creates a powerful way to solve problems.

In the chapters of this book, we decipher these concepts.

Contents

1. Introduction 1
- 1.1 *Mathematica* as a calculator 1
- 1.2 Numbers 5
- 1.3 Algebraic computations 9
- 1.4 Trigonometric computations 11
- 1.5 Variables 12
- 1.6 Equalities =, :=, == 13
- 1.7 Dynamic variables 15

2. Defining functions 19
- 2.1 Formulas as functions 19
- 2.2 Anonymous functions 23

3. Lists 25
- 3.1 Functions producing lists 28
- 3.2 Listable functions 32
- 3.3 Selecting from a list 34

4. Changing heads! 47

5. A bit of logic and set theory 54
- 5.1 Being logical 54
- 5.2 Handling sets 58
- 5.3 Decision making, If and Which 61

6. Sums and products 65
- 6.1 Sum 65

6.2 Product 70

7. Loops and repetitions 72
7.1 Do, For a While 72
7.2 Nested loops 81
7.3 Nest, NestList and more 84
7.4 Fold and FoldList 90
7.5 Inner and Outer 93

8. Substitution, *Mathematica* rules 96

9. Pattern matching 100

10. Functions with multiple definitions 112
10.1 Functions with local variables 118
10.2 Functions with conditions 119

11. Recursive functions 121

12. Linear algebra 127
12.1 Vectors 127
12.2 Matrices 128

13. Graphics 135
13.1 Two-dimensional graphs 135
13.2 Three-dimensional graphs 153

14. Calculus and equations 158
14.1 Solving equations 158
14.2 Calculus 163

15. Solutions to the Exercises 169

Further reading 184

Bibliography 185

Index 186

1 Introduction

This chapter introduces basic capabilities of *Mathematica*, which include simple arithmetic, handling algebraic and trigonometric expressions and assigning values to variables. We will also look at dynamic objects, allowing us to see changes in the variables as they happen.

In this chapter we give a quick introduction to the very basic things one can perform with Wolfram *Mathematica*®. We let the reader learn from reading the codes and avoid long and exhausting explanations, as the codes will speak for themselves.

1.1 *Mathematica* as a calculator

Mathematica can be used as a powerful calculator with the basic arithmetic operations; $+, -$ for addition and subtraction, $*, /$ for multiplication and division and ^ for powers.

```
10^9 - 987654321
12345679

2682440^4 + 15365639^4 + 18796760^4
18063007729216928108884849904 1

20615673^4
18063007729216928108884849904 1
```

The last two calculations show that

$$2682440^4 + 15365639^4 + 18796760^4 = 20615673^4,$$

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_1, © Springer-Verlag London Limited 2010

disproving a conjecture by Euler that three fourth powers can never sum to a fourth power.[1]

Mathematica can handle large calculations:

```
2^9941-1
```

```
34608828249085121524296039576741331672262866890023854779048928344500622080983411446436437554415370753366448674763505018641470709332373970608376690404229265789647993709760358469552319045484910050304149809818540283507159683562329419680597622813345447397208492609048551927706260549117935903890607959811638387214329942787636330953774381948448664711249676857988881722120330008214696844649561469971941269212843362064633138595375772004624402290646813260875582574884704893842439892702368849786430630930044229396033700105465953863020090730439444822025590974067005973057079950783296313093873988508019841625863519452291304256293667985958749572103117374779641889506070194171750600193715243003236363193426579851623604745120908986470743078036229830703819344548649375664799180425877557497383390331573508289102939235935275861718501994255834671861074548772439880729606244911940066680112823824095816458261761861746604034802056466823143718255492784779380991749580255263323265364577438941508489539699028185300578708762293298033382857354192282590221696026655322108347896020516865460114667379813060562474800550717182503337375022673073441785129507385943306843408026982289639865627325971753720872956490728302897497713583308679515087108592167432185229188116706374484964985490944305412774440794079895398574694527721321665808857543604774088429133272929486968974961416149197398454328358943244736013876096437505146992150326837445270717186840918321709483693962800611845937461435890688111902531018735953191561073191960711505984880700270887058427496052030631941911669221061761576093672419481606259890321279847480810753243826320939137964446657006013912783603230022674342951943256072806612601193787194051514975551875492521342643946459638539649133096977653332940182215800318288927807236860212898271030661811511896413189365784540029686001242039137696467018398359495411248456559731246073779877709207171067108245037074572201550158995917662449577680068024829766739203929954101642247764456712221498036579277084129255555428170455724308463899881299605192273139872912009020688206073376207589229947366640589742703581178687987569431507865442005560346962530939965395593231046643003914646580545296501404001942389755267555347682486246319514314931881709059725887801118502811905590736777711874328140886786742863021082751492584771012964518336519797173751709005056736459646963553313698192960002673895832892991267383457269803259989559975011766642010428885460856994644283419523294878748841059575019743878635311920421085580469246058253383296777194691145990192132498496881002118996828494133157316405630472548086892182344253819959038385241278684083347961141997010179297835565365075532913829865424622534682720750360674074595695812738374871782591852747316497058209518131290551924271028057302314555479362849901050929605584971237797898492183999703741589767415483070862914548472453672457262245013147999268168431046444943902225050485925083476189478888955252789840098819620001486857564023313650914562812719135485827508390789146997901942622488378946355 1
```

If a number of the form $2^n - 1$ happens to be prime, it is called a Mersenne prime. Recall that a *prime number* is a number which is divisible only by 1 and itself. It is easy to see $2^2 - 1$ and $2^3 - 1$ and $2^5 - 1$ are Mersenne primes. The list continues. In 1963, Gillies found that the above number, $2^{9941} - 1$, is a Mersenne prime. With my laptop it takes about 3 seconds for *Mathematica* to check that this is a prime number.[2]

```
PrimeQ[2^9941-1]
True
```

Back to easier calculations:

```
24/17
24/17
```

Mathematica always tries to give a precise value, thus gives back $\frac{24}{17}$ instead of attempting to evaluate the fraction.

[1] This conjecture remained open for almost 200 years, until Noam Elkies at Harvard came up with the above counterexample in 1988.

[2] The largest Mersenne prime found so far is $2^{43,112,609} - 1$ which was discovered in August 2008 at UCLA and has 12,978,189 digits.

```
Sin[Pi/5]
```

$$\frac{1}{2}\sqrt{\frac{1}{2}(5-\sqrt{5})}$$

Mathematica gives $\frac{1}{2}\sqrt{\frac{1}{2}(5-\sqrt{5})}$ as the value of $\sin(\pi/5)$, which is the precise equality. This shows *Mathematica* is not approaching the expressions numerically.

In order to get the numerical value, one can use the function `N`.

```
N[24/17]
1.41176

N[24/17, 20]
1.4117647058823529412

?N

N[expr] gives the numerical value of expr. N[expr, n] attempts to
give a result with n-digit precision.
```

As the above line shows, in order to get a quick description of a command, use `?Command`. If you need more explanation use `??Command`. If you need even more, use *Mathematica*'s Document Center in the Help Menu and search for the command. *Mathematica* comes with an excellent Help which contains explanations and many nice examples demonstrating how to use each of the functions available in this software.

All elementary mathematical functions are available here, `Log, Exp, Sqrt, Sin, Cos, Tan, ArcSin,...`. Here we evaluate

$$\sqrt{(\frac{\pi}{4})^2 + (0.5\,\mathrm{Log}[2])^2},$$

```
Sqrt[(Pi/4)^2 + (0.5 Log[2])^2]
0.858466
```

Problem 1.1

Use *Mathematica* to show that

$$\tan\frac{3\pi}{11} + 4\sin\frac{2\pi}{11} = \sqrt{11}.$$

$\Longrightarrow$ Solution.

```
Tan[3 Pi/11] + 4 Sin[2 Pi/11]
Cot[(5 Pi)/22] + 4 Sin[(2 Pi)/11]
```

We didn't get $\sqrt{11}$ as an answer. We ask *Mathematica* to try a bit harder.

```
?Simplify

Simplify[expr] performs a sequence of algebraic and other
transformations  on expr, and returns the simplest form it finds.
Simplify[expr,assum] does simplification using assumptions. >>


Simplify[Tan[3 Pi/11] + 4 Sin[2 Pi/11]]
Cot[(5 Pi)/22] + 4 Sin[(2 Pi)/11]

?FullSimplify

FullSimplify[expr] tries a wide range of transformations on expr
involving elementary and special functions, and returns the
simplest form it finds. FullSimplify[expr,assum] does
simplification using assumptions. >>

FullSimplify[Tan[3 Pi/11] + 4 Sin[2 Pi/11]]
Sqrt[11]
```

This problem introduces the commands `Simplify` and `FullSimplify`. If one is not happy with the result, one can always use `Simplify` and even `FullSimplify` to have *Mathematica* work a bit harder to come up with more simplification.

For a complete list of elementary functions have a look at Functional Navigator: Mathematics and Algorithms: Mathematical Functions in *Mathematica*'s Help.

Exercise 1.1

Show that $\sqrt{\sqrt[3]{64}\left(2^2 + (1/2)^2\right) - 1} = 4$.

Problem 1.2

Using *Mathematica*, explain why $4 + 6/4 * 3\hat{} - 2 + 1 = \dfrac{31}{6}$.

$\Longrightarrow$ Solution.

If you look at *Mathematica*'s Help, under "Special Ways to Input Expressions", you will see the following note: "The *Mathematica* language has a definite grammar which specifies how your input should be converted to internal form." One aspect of the grammar is that it specifies how pieces of your input should be grouped. The general rule is that if $\otimes$ has higher precedence than $\oplus$, then $a \oplus b \otimes c$ is interpreted as $a \oplus (b \otimes c)$, and $a \otimes b \oplus c$ is interpreted as $(a \otimes b) \oplus c$. You will then find a long table listing which operation has a higher precedence

and thus, based on that, you will be able to explain why $4+6/4*3\hat{}-2+1$ amounts to $\frac{31}{6}$.

However, common sense tells us that instead of creating an ambiguous expression such as $4+6/4*3\hat{}-2+1$, one should use parentheses () to group objects together and make the expression more clear. For example, one could write $4+\Big((6/4)*3\hat{}(-2)\Big)+1$, or even better use *Mathematica*'s Palette (Basic Math Assistance) and type

$$4+\frac{6}{4}\times 3^{-2}+1.$$

♣ TIPS

– Comments can be added to the codes using `(* comment *)`.

```
(* the most beautiful theorem in Mathematics *)
  E^(I Pi) + 1
  0
```

– The symbol `%` refers to the previous output produced. `%%` refers to the second previous output and so on.

– If in calculations you don't get what you are expecting, use `Simplify` or even `FullSimplify` (see Problem 1.1).

– To get the numerical approximation, use `N[expr]` or alternatively, `expr//N` (see Problem 2.2 for different ways of applying a function to a variable). Use `EngineeringForm[expr,n]` and `ScientificForm[expr,n]` to get other forms of numerical approximations to n significant digits.

– The mathematical constant e, the exponential number, is defined in *Mathematica* as `E`, or `Exp`. To get e^n use either `E^n` or `Exp[n]`.

1.2 Numbers

There are several standard ways to start with an integer number and produce new numbers out of it. For example, starting from 4, one can form $4\times 3\times 2\times 1$, which is represented by 4!.

```
4!
24

123!
121463043670253296757662432418812958554542170884833823153289 1
816182923589236216766883115696061264020217073583522129404778 2
591091570411651472186029519906261646730733907419814952960000
000000000000000000000000
```

The fundamental theorem of arithmetic states that one can decompose any number n as a product of powers of primes and this decomposition is unique, i.e., $n = p_1^{k_1} \cdots p_t^{k_t}$ where p_i's are prime. Thus $12 = 2^2 \times 3^1$ and $37534 = 2 \times 7^2 \times 383$. *Mathematica* can do all of these:

```
FactorInteger[12]
{{2, 2}, {3, 1}}

FactorInteger[37534]
{{2,1},{7,2},{383,1}}


2^1 * 7^2 * 383^1
37534

FactorInteger[6473434456376432]
{{2,4},{3239053,1},{124909859,1}}

PrimeQ[124909859]
True

Prime[8]
19
```

`Prime[n]` produces the n-th prime number. `PrimeQ[n]` determines whether n is a prime number. More than 2200 years ago Euclid proved that the set of prime numbers is infinite. His proof is used even today in modern books. However, it is not that long ago that we also learned that there is no simple formula that produces only prime numbers.

In 1640 Fermat conjectured that the formula $2^{2^n}+1$ always produces a prime number. Almost a hundred years later the first counterexample was found.

```
PrimeQ[2^(2^1)+1]
True

PrimeQ[2^(2^2)+1]
True

PrimeQ[2^(2^3)+1]
True
```

```
PrimeQ[2^(2^4)+1]
True

PrimeQ[2^(2^5)+1]
False

2^(2^5)+1
4294967297

FactorInteger[2^(2^5)+1]
{{641,1},{6700417,1}}
```

This shows that $2^{2^5}+1$ is not a prime number. In fact it decomposes into two prime numbers $2^{2^5}+1 = 641 \times 7600417$.

Problem 1.3

What is the probability that a randomly chosen 12-digit number will be a prime?

$\Longrightarrow$ Solution.

The probability is the number of 12-digit prime numbers over the number of all 12-digit numbers. So we start with finding how many 12-digit numbers exist:

```
10^13 - 10^12
9000000000000
```

Next, we will find how many 12-digit prime numbers exist. We will use the following built-in function of *Mathematica*.

```
?PrimePi

PrimePi[x] gives the number of primes less than
or equal to x. >>

PrimePi[10^13]
346065536839

PrimePi[10^12]
37607912018

N[(346065536839 - 37607912018)/9000000000000]*100
3.42731
```

So the probability that we randomly pick a 12-digit prime number is only 3.42 percent.

Exercise 1.2

Check that 123456789098765432111 is prime.

Exercise 1.3

Check that for any given positive integer $n \geq 3$, the least common multiple of the numbers $1, 2, \cdots, n$ is greater than 2^{n-1}. (Hint: see `LCM`.)

Exercise 1.4

Check that the number 32! ends with 7 zeros.

Problem 1.4

The function `Mod[m,n]` gives the remainder on division of m by n. Let $a = 98$ and $b = 75$. Show that $a^a b^b$ ends with exactly 98 zeros.

$\Longrightarrow$ Solution.

Clearly you can directly compute $a^a b^b$ and count the number of zeros as in Exercise 1.4. But this time you have to count whether the number ends with 98 zeros. Here is a more clever way: Check whether the remainder of this number when divided by 10^{98} is zero, but its remainder on division by 10^{99} is not.

```
Mod[98^98 75^75, 10^98]
0

Mod[98^98 75^75, 10^99]
500000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000
```

Problem 1.5

Recall that for integers m and n, the binomial $\binom{n}{m}$ is defined as $\dfrac{n!}{m!(n-m)!}$. Using *Mathematica*, check that

$$\binom{m+n}{m} = \frac{(m+n)!}{m!n!}.$$

$\Longrightarrow$ Solution.

Not only is the binomial function available in *Mathematica*, but also *Mathematica* can perfectly handle it symbolically as the solution to this problem shows. We will talk more about symbolic computations in Section 1.3.

```
Binomial[m + n, n] == (n + m)!/(n! m!)
Binomial[m + n, n] == (m + n)!/(m! n!)

FullSimplify[Binomial[m + n, n] == (n + m)!/(n! m!)]
True
```

This is another instance where we need to use `FullSimplify` to make *Mathematica* work harder to come up with the result.

We will discuss the different equalities available in *Mathematica* in Section 1.6. However, for the time being, note that `==` is used to compare both sides of equations.

There are several more integer functions available in *Mathematica*, which can be found in Functional Navigator: Mathematics and Algorithms: Mathematical Functions: Integer Functions.

♣ TIPS

– The command `NextPrime[n]` gives the next prime larger than `n` and `PrimePi[n]` gives the number of primes less than or equal to `n` (see Problem 1.3).

– For integers m and n, one can find unique numbers q and r such that r is positive, $m = qn + r$ and $r <\mid q \mid$. Then `Mod[m,n]=r` and `Quotient[m,n]=q`.

– If an evaluation is taking a long time, in order to stop the evaluation use Alt+. (for Windows) and Cmd+. (for Apple Macintosh). For example try to calculate the 1234567891011-th prime number. If you can't wait to get the result, you now know how to stop the process. There are cases where pressing Alt+. does not help, even if you do this several times. In these situations, use the Evaluation menu and choose Quit Kernel.

1.3 Algebraic computations

One of the abilities of *Mathematica* is to handle symbolic computations, i.e., *Mathematica* can comfortably work with symbols (we have seen one example of this in Problem 1.5). Consider the expression $(x + 1)^2$. One can use *Mathematica* to expand this expression:

```
Expand[(1 + x)^2]
1 + 2 x + x^2
```

Mathematica can also do the inverse of this task, namely to factorize an expression:

```
Factor[1 + 2 x + x^2]
(1 + x)^2
```

While expansion of an algebraic expression is a simple and routine procedure, the factorization of algebraic expressions is often quite challenging. My favorite example is this one. Try to factorize the expression $x^{10} + x^5 + 1$. Here is one way to do that:

$$
\begin{aligned}
& x^{10} + x^5 + 1 \quad \text{(adding } x^i - x^i, 1 \le i \le 9, \text{ to the expression we have)} \\
&= x^{10} + \underbrace{x^9 - x^9} + \underbrace{x^8 - x^8} + \cdots + \underbrace{x^6 - x^6} + \\
&\underbrace{+x^5 - x^5} + x^5 + \underbrace{x^4 - x^4} + \cdots + \underbrace{x - x} + 1 \quad \text{(now rearranging the terms)} \\
&= x^{10} + x^9 + x^8 - x^9 - x^8 - x^7 + x^7 + x^6 + x^5 - x^6 - x^5 - x^4 \\
&\qquad + x^5 + x^4 + x^3 - x^3 - x^2 - x + x^2 + x + 1 \\
&= x^8(x^2 + x + 1) - x^7(x^2 + x + 1) + x^5(x^2 + x + 1) - x^4(x^2 + x + 1) \\
&\qquad + x^3(x^2 + x + 1) - x(x^2 + x + 1) + x^2 + x + 1 \\
&= (x^2 + x + 1)(x^8 - x^7 + x^5 - x^4 + x^3 - x + 1)
\end{aligned}
$$

Mathematica can easily come up with this factorization:

```
Factor[x^10 + x^5 + 1]
(1 + x + x^2) (1 - x + x^3 - x^4 + x^5 - x^7 + x^8)
```

It is a fact that the product of four consecutive numbers plus one is always a squared number; here is a proof:

```
Factor[n (n + 1) (n + 2) (n + 3) + 1]
(1 + 3 n + n^2)^2
```

♣ TIPS

– The command `Together` makes a sum of terms into one single term over a common denominator. The command `Apart` does the (almost) reverse of `Together` (see Exercise 1.6).

Exercise 1.5

Factorize the polynomial $(1 + x)^{30} + (1 - x)^{30}$.

Exercise 1.6

Using `Together`, write the expression

$$\frac{1}{1 + x} + \frac{1}{1 + \frac{1}{1+x}}$$

with one single denominator. Now apply `Apart` to the result to get an expression as a sum of terms with minimal denominators.

There are several more algebraic functions available in *Mathematica*, which can be found in Functional Navigator: Mathematics and Algorithms: Mathematical Functions: Polynomial Algebra.

1.4 Trigonometric computations

Similar to algebraic expressions (Section 1.3), *Mathematica* can handle trigonometric expressions. Here one uses `TrigExpand` and `TrigFactor` to work with trig. expressions. Let us start with the best-known trig. identity, $\cos^2(x) + \sin^2(x) = 1$.

```
Cos[x]^2 + Sin[x]^2
Cos[x]^2 + Sin[x]^2

Simplify[Cos[x]^2 + Sin[x]^2]
1

Expand[Cos[x]^2 + Sin[x]^2]
Cos[x]^2 + Sin[x]^2

TrigExpand[Cos[x]^2 + Sin[x]^2]
1
```

Mathematica is quite at ease with trig. identities as the following problem demonstrates.

Problem 1.6

Using *Mathematica*, check that the following trigonometric identities hold:

$$\sin^3(x)\cos^3(x) = \frac{3\sin(2x) - \sin(6x)}{32}$$

$$\frac{1 + \sin(x) - \cos(x)}{1 + \sin(x) + \cos(x)} = \tan(x/2)$$

$\Longrightarrow$ Solution.

The only challenge here is to translate these expressions correctly into *Mathematica*.

```
Simplify[Sin[x]^3 Cos[x]^3 == (3 Sin[2 x] - Sin[6 x])/32]
True

Simplify[(1 + Sin[x] - Cos[x])/(1 + Sin[x] + Cos[x]) == Tan[x/2]]
True
```

Note that `==` is used to compare both sides of equations. We will discuss the different equalities available in *Mathematica* in Section 1.6.

Exercise 1.7

Using *Mathematica*, observe that

$$\frac{1+\sin(x)-\cos(x)}{1+\sin(x)+\cos(x)} = \tan(x/2).$$

♣ TIPS

– The argument of trig. functions, e.g., `Sin`, is assumed to be in radians. (Multiply by `Degree` to convert from degrees to radians.)

```
Sin[30 Degree]
1/2
```

1.5 Variables

In order to feed data into a computer program one needs to define variables to be able to assign data to them. As long as you use common sense, any names you choose for variables are valid in *Mathematica*. Names like `x, y, x3, myfunc, xQuaternion,...` are all fine. Do not use underscore `_` to define a variable.[3] Also note that *Mathematica* is case sensitive, thus `xy` and `xY` are considered as two different variables.

```
x = 3
3

y = 4
4

x^2 + y^2
25

Sqrt[x^2 + y^2]
5
```

If we need to enter several statements in one line, we can separate them with `;`.

[3] This is quite common in Pascal or C, to define variables such as `x_printer, com_graph,...`. In *Mathematica*, the underscore is reserved and will be used in the definition of functions in Chapter 2.

```
t = 7; s = 4; t!/(s! (t - s)!)
35
```

Problem 1.7

Using `Expand`, for `Expand[(1+x)^ 2]`, instead of obtaining $1 + 2x + x^2$ we get

```
Expand[(1 + x)^2]
16
```

What seems to be the problem?

$\Longrightarrow$ SOLUTION.

If you are working through this section, in the beginning of this section you have already defined `x=3`. Thus *Mathematica* will take this into account when working with the expression $(1 + x)^2$, which then amounts to 16. This shows one of the common mistakes one tends to make in *Mathematica*, namely using variables which have already been defined, as undefined symbols. In order to clear the value or definition of a variable, use `Clear`.

```
Clear[x]

Expand[(1 + x)^2]
1 + 2 x + x^2
```

♣ TIPS

- Use `Clear[x]` to clear the value given to the variable `x`, before using `x` as a symbol.
- Use `Clear["Global‘*"]` to clear values and definitions given to *all the* symbols.
- Assigning a value to a symbol works globally. That means, if you open a new NoteBook, the values given to variables in a previous NoteBook still exist.

1.6 Equalities =, :=, ==

Primarily there are three equalities in *Mathematica*, =, := and ==. There is a fundamental difference between = and :=. Study the following example:

```
x=5;y=x+2;

y
7

x=10
10

y
7

x=15
15

y
7
```

So changing the value of x does not affect the value of y. Now compare this with the following one, when we replace = with := in the definition of y.

```
x=5;y:=x+2;

y
7

x=10
10

y
12

x=15
15

y
17
```

From the first example it is clear that when we define y=x+2 then y takes the *value* of x+2 and this will be assigned to y. No matter if x changes its value, the value of y remains the same. In other words, y is independent of x. But in y:=x+2, y is dependent on x, and when x changes, the value of y changes too. Namely using := makes y a function with variable x. The following is an excellent example to show the difference between = and :=.

```
?Random

Random[ ] gives a uniformly distributed pseudorandom Real in the
range 0 to 1.

x=Random[]
0.246748

x
0.246748
```

```
x
0.246748

x:=Random[]

x
0.60373
x
0.289076

x
0.564378
```

When defining `x=Random[]`, the function `Random` generates a number and this number will be assigned to `x`. Each time we call on `x`, this number is what we get. However, when we define `x:=Random[]`, then the definition of `x` is `Random[]`. Thus when we call `x`, we have in fact called on `Random` which then generates a new random number.

We will examine this difference between `=` and `:=` again in Example 3.5.

Finally, the equality == is used to compare:

```
5==5
True

3==5
False
```

We will discuss this further in Section 5.1.

1.7 Dynamic variables

The new version of *Mathematica*[4] comes with an ability to define *dynamic* variables. This means one can monitor the changes in a variable "live", i.e., as they happen. We are going to introduce this feature early in the book to take advantage of it as we go along.

We saw in Section 1.5 that one can define variables and assign values to them.

```
x = 3
3

x = 10
10
```

Here when we assign 10 to `x`, although this is the new value of `x`, in the line above it, i.e., `x=3, 3` does not change. However, if we define the variable `x` as

[4] Currently version 7.

a dynamic variable, then each time we change the value of `x` anywhere in the program, all the old values also change to the new value accordingly.

```
Dynamic[x]
10
```

Then if in the next line we change the value to `x=15`, we will see that the value of the previous line immediately changes to 15 as well.

```
Dynamic[x]
15

x=15
15
```

One can control the value of the variable `x` by introducing a *slider*.

```
Slider[Dynamic[x]]
```

```
Dynamic[x]

0.326
```

You will see that as you drag the slider, the value of `x` changes. This already gives us a lot of power, as the following example will show. Recall from Section 1.3 that we can expand expressions using `Expand`.

```
Expand[(1 + y)^2]
1 + 2 y + y^2

Expand[(1 + y)^3]
1 + 3 y + 3 y^2 + y^3
```

Now we can simply consider $(1 + y)^n$ and then, defining n as a dynamic variable and controlling it with a slider, we can change the value of n by dragging the slider and see the expansions of $(1 + y)^n$ for different values of n as they happen right in front of our eyes!

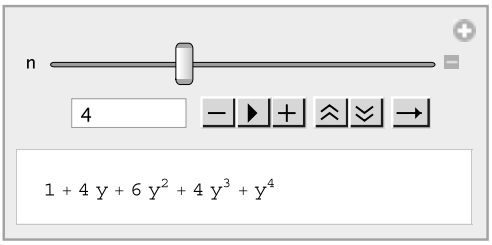
```
Slider[Dynamic[n], {1, 10, 1}]
```

```
Dynamic[Expand[(1 + y)^n]]

1 + 7 y + 21 y^2 + 35 y^3 + 35 y^4 + 21 y^5 + 7 y^6 + y^7
```

Note that, when defining `Slider`, the value of x varies from 0 to 1. If we want to change this interval, as in the previous example, we can specify the interval and the step that is added to x each time by using {`xmin,xmax, step`}.

A similar concept to `Slider` is the function `Manipulate` which allows us to change the value of a variable and see the result "live".

```
Manipulate[Expand[(1 + y)^n], {n, 1, 10, 1}]
```



The control buttons can be used to start or stop the process, and make it faster or slower. Just try it out.

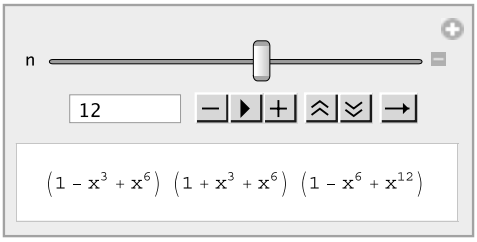
We will see later, for example in Chapter 13 when we are dealing with graphics, that we can use `Manipulate` to change the value of our parameters and see how the graph changes accordingly.

Problem 1.8

Using `Manipulate`, observe that the polynomial $x^{2n}+x^n+1$ can be decomposed into smaller factors for any $1 \leq n \leq 20$ except $n = 1, 3, 9$.

$\Longrightarrow$ Solution.

```
Manipulate[Factor[x^(2 n) + x^n + 1], {n, 1, 20, 1}]
```



Problem 1.9

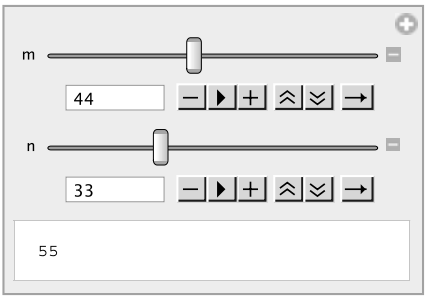
Using `Manipulate`, find out for which positive integers n and m, between 1 and

100, $m^2 + n^2$ is a squared number (these are called Pythagorean pairs).

$\Longrightarrow$ Solution.

Later in Chapter 7, when we are discussing loops, we will write a program to generate these numbers (Problem 7.11). Here we will use `Manipulate`, defining two dynamic variables m and n, and we will look at the result of $\sqrt{m^2 + n^2}$, and when this is an integer then (m, n) is a Pythagorean pair.

```
Manipulate[Sqrt[m^2 + n^2], {m, 1, 100, 1}, {n, 1, 100, 1}]
```



2
Defining functions

This chapter shows how to define functions in *Mathematica*. Examples of functions with several variables and anonymous functions are given.

Functions in mathematics define rules about how to handle data. For example, the function f defined as $f(n) = n^2 + 1$ will get as an input (a number) n and its output will be $n^2 + 1$. Besides the wide use of $f(n)$, there are several other ways to show how the rule f applies to n, such as $n \xrightarrow{f} n^2 + 1$, $nf = n^2 + 1$ or even $n^f = n^2 + 1$. We will see that Wolfram *Mathematica*®, besides supporting `f[n]`, has two other ways to apply a function to data, namely `f@n` and `n//f`.

2.1 Formulas as functions

Defining functions is one of the strong features of *Mathematica*. One can define a function in several different ways in *Mathematica* as we will see in the course of this chapter.

Let us start with a simple example of defining the formula $f(n) = n^2 + 4$ as a function and calculating $f(-2)$:

```
f[n_]:= n^2 +4
```

First notice that in defining a function we use :=. The symbol `n` is a dummy variable and, as expected, one plugs in the data in place of `n`.

```
f[-2]
8
```

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_2, © Springer-Verlag London Limited 2010

In fact as we will see later, one can plug "anything" in place of `n` and that is why functions in *Mathematica* are far superior to those in `C` and other languages.

```
f[3.56]
16.6736

f[elephant]
4 + elephant^2
```

One more note about the extra underscore `_` in the definition of the function. The underscore which will be called blank here, stands (or rather sits) for the expression which will be passed to `f`. So we cheated a little when we said we plug data in place of `n`. The underscore, named `n`, gets the data and `f` applies its rule to this data. This data can have any pattern. If this is confusing, forget this technicality now. We will talk about patterns and pattern matching in Chapter 9 and leave it as it is for the moment.

We proceed by defining the function $g(x) = x + \sin(x)$.

```
g[x_]:= x+Sin[x]
g[Pi]
```

π

One can define functions of several variables. Here is a simple example defining $f(x, y) = \sqrt{x^2 + y^2}$.

```
f[x_,y_]:=Sqrt[x^2+y^2]
f[3,4]
5
```

It is very easy to compose functions in *Mathematica*, i.e., apply functions one after the other on data. Here is an example of this:

```
f[x_]:=x^2+1
g[x_]:=Sin[x]+Cos[x]

f[f[x]]
1 + (1 + x^2)^2

f[g[x]]
1+(Cos[x]+Sin[x])^2

g[f[x]]
Cos[1+x^2]+Sin[1+x^2]
```

This example clearly shows that the composition of functions is not a commutative operation, that is $fg \neq gf$.

Problem 2.1

Using *Mathematica*, show that

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}} = \frac{3+2x}{5+3x}. \tag{2.1}$$

$\Longrightarrow$ Solution.

If we define $f(x) = \frac{1}{1+x}$, then $f(f(x)) = \frac{1}{1+\frac{1}{1+x}}$ and $f(f(f(x))) = \frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}$. This shows a way to capture the left-hand side of Equality 2.1 without going through the pain of typing it.

```
f[x_] := 1/(1 + x)

f[f[f[f[x]]]]
1/(1 + 1/(1 + 1/(1 + 1/(1 + x))))

Simplify[f[f[f[f[x]]]]]
(3 + 2 x)/(5 + 3 x)
```

Exercise 2.1

Define $f(x) = \sqrt{1+x}$ in *Mathematica* and show that

$$f(f(f(f(f(x))))) = \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}}.$$

Recall that the Fibonacci sequence starts with the sequence, $1, 1$ and the next term in the sequence is the sum of the previous two numbers. Thus the first 7 Fibonacci numbers are $1, 1, 2, 3, 5, 8, 13$. The function `Fibonacci[n]` gives the n-th Fibonacci number.

```
Fibonacci[7]
13
```

And this is a little function to find out if the n-th Fibonacci number is divisible by 5 (see Problem 1.4 for the use of function `Mod`).

```
remain[n_]:=Mod[Fibonacci[n],5]
remain[14]
2

remain[15]
0
```

Thus the 15th Fibonacci number is divisible by 5. Note that the function `remain` is itself a composition of two functions, namely the functions `Fibonacci` and `Mod`.

Besides the traditional way of `remain[x]`, there are two other ways to apply a function to an argument as follows:

```
15//remain
0

remain@15
0
```

Problem 2.2

Design a function to check whether for a number n, the formula $n!+1$ generates a prime number.

$\Longrightarrow$ Solution.

The function is a composition of two functions, first to generate `n!+1` and then using `PrimeQ` to test whether this number is prime.

```
pTest[n_] := PrimeQ[n! + 1]

pTest[2]
True

pTest[3]
True

pTest[4]
False
```

Here are the other ways to apply a function to a variable.

```
4//pTest
False

pTest@4
False
```

Problem 2.3

Define the function

$$b(n) = 1 + \binom{n}{1} + \binom{n}{2} + \binom{n}{3},$$

and find out whether 2^{2008} is divisible by $b(23)$.

$\Longrightarrow$ Solution.
Recall that $\binom{n}{m}$ is defined by `Binomial[n,m]` in *Mathematica*.

```
b[n_] := Binomial[n, 1] + Binomial[n, 2] + Binomial[n, 3] + 1

b[23]
2048

Mod[2^2008, b[23]]
0
```

Recall from Problem 1.4 that `Mod[m,n]` returns the remainder on division of `m` by `n`. If this remainder is zero, it clearly means that `m` is divisible by `n`. There is also the command `Divisible` which takes care of this situation.

```
?Divisible

Divisible[n,m] yields True if n is divisible by m, and yields
False if it is not. >>

Divisible[2^2008, b[23]]
True
```

Later, in Problem 3.4, we will determine all the positive integers n between 3 and 50 for which 2^{2008} is divisible by $b(n)$ and will see that in fact there are very few n with this property.

Later, in Chapter 10, we will define functions with conditions, functions with several definitions and functions containing several lines of code (a procedure).

2.2 Anonymous functions

Sometimes we need to "define a function as we go" and use it on the spot. *Mathematica* enables us to define a function without giving it a name (nor any reference to any specific variables) use it, and then move on! These functions are called *anonymous* or *pure* functions. Obviously if we need to use a specific function frequently, then the best way is to give it a name and define it as we did in Section 2.1. Here is an anonymous function equivalent to $f(x) = x^2 + 4$:

```
(#^2+4)&
```

The expression `(#^2+4)&` defines a nameless function. As usual we can plug in data in place of `#`. The symbol `&` determines where the definition of the function is completed.

```
(#^2+4)&[5]
 29
```

Compare the following:

```
r[x_] := x (x + 1)

r[4]
20

# (# + 1) &[4]
20

4 // # (# + 1) &
20
```

Exercise 2.2

What do the following pure functions do?

```
Fibonacci[15]//Mod[#,5]&

PrimeQ[#! + 1] &@4

18//2^#+#&
```

Anonymous functions can handle several variables. Here is an example of an anonymous function for $f(x, y) = \sqrt{x^2 + y^2}$.

```
Sqrt[#1^2+#2^2]&[3,4]
5
```

As you might guess, `#1` and `#2` refer to the first and second variables in the function.

3
Lists

A list is a collection of data. In this chapter we study how to handle lists and have access to the elements in the lists. Functions which generate lists are studied and methods of selecting elements from a list with a specific property are examined. Get ready for some serious programming!

One can think of a computer program as a function which accepts some (crude) data or information and gives back the data we would like to obtain. *Lists* are the way Wolfram *Mathematica*® handles information. Roughly speaking, a list is a collection of objects. The objects could be of any type and pattern. Let us start with an example of a list:

```
{1, -4/7, stuff, 1 - 2 x + x^2}
```

This looks like a mathematical set. One difference is that lists respect order:

```
{1, 2} == {2, 1}
False
```

The other difference is that a list can contain a copy of the same object several times:

```
{1,2,1} == {1,2}
False
```

The natural thing here is to be able to access the elements of a list. Let us define a list `p` as follows:

```
p = {x, 1, -8/3, a, b, {c, d, e}, radio}
{x, 1, -(8/3), a, b, {c, d, e}, radio}
```

As this example shows, the list `p` can have any sort of data, even another list as one of its elements. Here is how we can access the elements of the list:

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_3, © Springer-Verlag London Limited 2010

```
p[[1]]
x

p[[5]]
b

p[[-1]]
radio

p[[{2, 4}]]
{1, a}

p[[{-2, 5}]]
{{c, d, e}, b}

p[[-2, {2, 3}]]
{d, e}
```

Examining the above examples reveals that `p[[i]]` gives the i-th member of the list. There are other commands to access the elements of a list as follows:

```
p = {x, 1, -8/3, a, b, {c, d, e}, radio}
{x, 1, -(8/3), a, b, {c, d, e}, radio}

First[p]
x

Last[p]
radio

Drop[p, 3]
{a, b, {c, d, e}, radio}

Take[p, 2]
{x, 1}

Rest[p]
{1, -(8/3), a, b, {c, d, e}, radio}

Rest[%]
{-(8/3), a, b, {c, d, e}, radio}

Rest[%]
{a, b, {c, d, e}, radio}

Most[p]
{x, 1, -(8/3), a, b, {c, d, e}}

Most[Rest[p]] == Rest[Most[p]]
True
```

Most of these commands are self-explanatory and a close look at the above examples shows what each of them will do (see the Tips on page 5 for `%`). All these commands and more are listed in the *Mathematica* Help under Functional Navigator: Core Language: List Manipulation: Elements of Lists.

Problem 3.1

Let `p={a,b,{c,d},e}`. From this list produce the list `{a,b,c,d,e}`.

$\Longrightarrow$ Solution.

```
{p[[1]], p[[2]], p[[3, 1]], p[[3, 2]], p[[4]]}
{a, b, c, d, e}
```

One can also use the command `Flatten` to get rid of extra { and }. We will discuss this command in Section 7.2, where we will be looking at nested loops which create nested lists.

```
p = {a, b, c, {d, e}, f}
{a, b, c, {d, e}, f}

Flatten[p]
{a, b, c, d, e, f}
```

One of the secrets of writing codes comfortably is that one should be able to manipulate lists easily. Often in applications, situations like the following arise:

– Given $\{x_1, x_2, \cdots, x_n\}$ and $\{y_1, y_2, \cdots, y_n\}$, produce

$$\{x_1, y_1, x_2, y_2, \cdots, x_n, y_n\}.$$

– Given $\{x_1, x_2, \cdots, x_n\}$ and $\{y_1, y_2, \cdots, y_n\}$, produce

$$\{x_1 + y_1, x_2 + y_2, \cdots, x_n + y_n\}.$$

– Given $\{x_1, x_2, \cdots, x_n\}$ and $\{y_1, y_2, \cdots, y_n\}$, produce

$$\big\{\{x_1, y_1\}, \{x_1, y_2\}, \{x_1, y_3\}, \cdots, \{x_1, y_n\}$$
$$\{x_2, y_1\}, \{x_2, y_2\}, \cdots, \{x_n, y_1\}, \{x_n, y_2\}, \cdots, \{x_n, y_n\}\big\}.$$

– Given $\{x_1, x_2, \cdots, x_n\}$, produce

$$\{x_1, x_1 + x_2, \cdots, x_1 + x_2 + \cdots + x_n\}.$$

– Given $\{x_1, x_2, \cdots, x_n\}$, produce

$$\Big\{\big\{\{x_1\}, \{x_2, \ldots, x_n\}\big\}, \big\{\{x_1, x_2\}, \{x_3, \ldots, x_n\}\big\} \ldots$$
$$\big\{\{x_1, \ldots x_{n-1}\}, \{x_n\}\big\}\Big\}.$$

Mathematica provides us with commands to obtain the above arrangements easily. We will look at these commands in Sections 7.4, 7.5, 12.1 and Problem 9.4.

♣ TIPS

– One can add elements to a list. There are several commands to handle this, including `Append`, `Prepend` and `Insert`.

```
?Insert

Insert[list,elem,n] insert elem at position n in list. If n is
negative, the position is counted from the end.

Insert[{i, think, i, am}, "therefore", 3]
{i, think, therefore, i, am}
```

See also Example 5.3 for the use of `Append`.

– Commands `Sort`, `Reverse`, `RotateLeft` and `RotateRight` are available to rearrange the order of a list.

– Commands `Delete` and `Drop` are available to remove elements from a list.

3.1 Functions producing lists

Mathematica provides us with commands of which the output is a list. These commands have a nature of repetition and replace loops in procedural programming (more on this in Chapter 7). Let us look at some of them here before starting to write more serious codes.

```
Range[10]
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Range[3, 11]
{3, 4, 5, 6, 7, 8, 9, 10, 11}

Range[2, 17, 4]
{2, 6, 10, 14}

?Range

Range[imax] generates the list {1,2,...,imax}.
Range[imin, imax] generates the list {imin,...,imax}.
Range[imin,imax,di] uses step di.>>
```

One of the most useful commands which produces a list is `Table`.

```
Table[2 n + 1, {n, 1, 13}]
{3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27}
```

In `Table[2n+1,{n,1,13}]`, `n` runs from 1 to 13 and each time the function `2n+1` is evaluated.

The following example shows how easily we can work symbolically in *Mathematica*.

```
Table[x^i + y^i, {i, 2, 17, 4}]
{x^2 + y^2, x^6 + y^6, x^10 +y^10, x^14 + y^14}
```

As in the last example of `Range`, here in `Table`, `i` starts from 2 with steps 4 and thus takes the values `2,6,10,14`.

Here is one more example demonstrating how beautifully *Mathematica* can handle symbols

`Table[`x_i`,{i,1,10}]`

$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$

Problem 3.2

Produce the list of the first 30 Fibonacci numbers.

$\Longrightarrow$ Solution.

All we have to do is to let `i` run from 1 to 30 and each time plug `i` into the function `Fibonacci[i]` which produces the i-th Fibonacci number. This can be done with `Table` as follows:

```
Table[Fibonacci[i], {i, 1, 30}]
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610,
987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025,
121393, 196418, 317811, 514229, 832040}
```

Problem 3.3

Find all natural numbers n between 1 and 15 for which the polynomial $x^n + 64$ can be written as a product of two nonconstant polynomials with integer coefficients.

$\Longrightarrow$ Solution.

We have seen that `Factor` will give a factorization of an expression (if it is possible).

```
Table[Factor[x^i + 64], {i, 1, 10}]
{64 + x, 64 + x^2, (4 + x) (16 - 4 x + x^2),
(8 - 4 x + x^2) (8 + 4 x + x^2),  64 + x^5,
(4 + x^2) (16 - 4 x^2 + x^4),  64 + x^7,
(8 - 4 x^2 + x^4) (8 + 4 x^2 + x^4),
(4 + x^3) (16 - 4 x^3 + x^6), 64 + x^10}
```

Although this solves the problem (for $1 \leq n \leq 10$), the output is not arranged nicely and it is hard to understand the answer. It would be easier if we keep track of `i` as well. We change the code slightly. For this we need `Print`

```
?Print
Print[expr] prints expr as output. >>
```

Here is the improved code:

```
Table[Print[i, "  ", Factor[x^i + 64]], {i, 1, 15}];

1  64+x

2  64+x^2

3  (4+x) (16-4 x+x^2)

4  (8-4 x+x^2) (8+4 x+x^2)

5  64+x^5

6  (4+x^2) (16-4 x^2+x^4)

7  64+x^7

8  (8-4 x^2+x^4) (8+4 x^2+x^4)

9  (4+x^3) (16-4 x^3+x^6)

10  64+x^10

11  64+x^11

12  (2-2 x+x^2) (2+2 x+x^2) (4-4 x+2 x^2-2 x^3+x^4)
(4+4 x+2 x^2+2 x^3+x^4)

13  64+x^13

14  64+x^14

15  (4+x^5) (16-4 x^5+x^10)
```

From the list, for $n = 3, 4, 6, 8, 9, 12, 15$ we have a factorization into two nonconstant polynomials (in fact, for $n = 3k$ and $4k$ the polynomial can be factorized into two polynomials).

Problem 3.4

Determine all the positive integers n between 3 and 50 for which 2^{2008} is divisible by

$$1 + \binom{n}{1} + \binom{n}{2} + \binom{n}{3}.$$

$\Longrightarrow$ Solution.

Recall that if `Mod[m,n]` returns zero, then `m` is divisible by `n`. We first define a function $b(n) = 1 + \binom{n}{1} + \binom{n}{2} + \binom{n}{3}$ (see Problem 2.3) and then use a `Table` to check when 2^{2008} is divisible by $b(n)$ for different n.

```
b[n_] := Binomial[n, 1] + Binomial[n, 2] + Binomial[n, 3] + 1

Table[Mod[2^2008, b[n]], {n, 3, 50}]

{0, 1, 16, 16, 0, 70, 16, 80, 168, 133, 268, 316, 448, 256,
706, 796, 1096, 723, 1092, 1030, 0, 256, 458, 1240, 2704, 2604,
606, 2922, 640, 1664, 2704, 4076, 2824, 1936, 1024, 8336, 256,
7882, 6974, 4192, 4568, 6061, 1076, 6896, 704, 16669, 6856,
16032}
```

It is clear from the list that only for $n = 3, 7$ and 23, is 2^{2008} divisible by the above expression (in fact, one can show that these are the only numbers with this property).

Here is a nice example showing the difference between the two equalities = and :=.

Example 3.5

This example uses `BarChart` which is a graphic function. The example is based on the discussion in Section 1.6.

```
?BarChart
BarChart[{y1,y2,...}] makes a bar chart with bar lengths y1,y2,...

x=RandomInteger[{1,1000}];
BarChart[Table[x,{200}]]


x:=RandomInteger[{1,1000}]
BarChart[Table[x,{200}]]
```

In order to understand this example better, get the list generated by `Table[x,1000]` for each of the definitions of `x` individually and compare them.

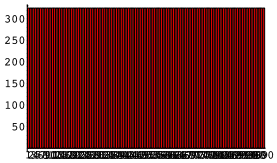


Figure 3.1 Using = as equality

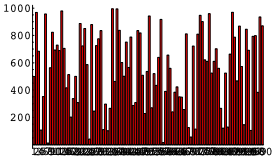


Figure 3.2 Using := as equality

3.2 Listable functions

There are times when we would like to apply a function to all the elements of a list. Suppose `f` is a function and {`a,b,c`} is a list. We want to be able to "push" the function `f` inside the list and get {`f[a],f[b],f[c]`}. Many of *Mathematica*'s built-in functions have the property that they simply "go inside" a list. This property of a function is called *listable*. For example `Sqrt` is a listable function.

```
Sqrt[{a, b, c, d, e}]
```

$$\{\sqrt{a}, \sqrt{b}, \sqrt{c}, \sqrt{d}, \sqrt{e}\}$$

All the arithmetic functions are listable:

```
1+ {a, b, c, d, e}
{1 + a, 1 + b, 1 + c, 1 + d, 1 + e}

{a, b, c, d, e}^3
{a^3,b^3,c^3,d^3,e^3}

1/{a, b, c, d, e}
{1/a,1/b,1/c,1/d,1/e}
```

We will use the function `Sqrt` in the following, to show that the product of four consecutive numbers plus one is always a squared number.

```
Table[n (n + 1) (n + 2) (n + 3) + 1, {n, 1, 10}]
{25, 121, 361, 841, 1681, 3025, 5041, 7921, 11881, 17161}

Sqrt[%]
{5, 11, 19, 29, 41, 55, 71, 89, 109, 131}
```

Not all the functions are listable. If we want to push a function into a list, the command to use is `Map`.

```
f[{a, b, c, d, e}]
f[{a, b, c, d, e}]

Map[f, {a, b, c, d, e}]
{f[a], f[b], f[c], f[d], f[e]}
```

The equivalent shorthand to apply a function to a list is `/@` as follows:

```
f /@ {a, b, c, d, e}
{f[a], f[b], f[c], f[d], f[e]}
```

Here is an example:

```
Table[(1 + x)^i, {i, 5}]
{1 + x, (1 + x)^2, (1 + x)^3, (1 + x)^4, (1 + x)^5}

Expand /@ %
{1 + x, 1 + 2 x + x^2, 1 + 3 x + 3 x^2 + x^3,
 1 + 4 x + 6 x^2 + 4 x^3 + x^4,
 1 + 5 x + 10 x^2 + 10 x^3 + 5 x^4 + x^5}
```

Problem 3.6

The formula $n^2 + n + 41$ has a very interesting property. Observe that this formula produces prime numbers for all n between 0 and 39.

$\Longrightarrow$ SOLUTION. First we produce the numbers:

```
Table[n^2 + n + 41, {n, 0, 40}]
{41, 43, 47, 53, 61, 71, 83, 97, 113, 131, 151, 173, 197,
223, 251, 281, 313, 347, 383, 421, 461, 503, 547, 593, 641, 691,
743, 797, 853, 911, 971, 1033, 1097, 1163, 1231, 1301, 1373, 1447,
1523, 1601, 1681}
```

Then we apply `PrimeQ` to this list. This function is listable.

```
PrimeQ[%]
{True, True, True, True, True, True, True, True, True, True,
True, True, True, True, True, True, True, True, True, True,
True, True, True, True, True, True, True, True, True, True,
True, True, True, True, True, True, True, True, False}
```

The list shows that for n between 0 and 39, the above formula produces prime numbers, however, for $n = 40$, the last number in the list $40^2 + 40 + 41$ is not prime. This formula was found by Euler around 1772. One wonders, if one changes 41 to another number in the formula $n^2 + n + 41$, whether one gets more consecutive prime numbers. We will examine this in Problem 7.2.

Of course we could also simply write

```
Table[PrimeQ[n^2 + n + 41], {n, 0, 40}]
```

as well. As we said, there are many ways to approach a problem in *Mathematica*.

Exercise 3.1

Generate 6 random integers n between 1 and 3095 and show that $n^6 + 1091$ is not prime. (Hint: see `RandomInteger` for generating random numbers.)

3.3 Selecting from a list

So far we have been able to produce a list of data by using functions producing lists. The next step is to be able to choose, from a list, certain data which fit a specific description. This can be achieved by the command `Select` as the following problem demonstrates.

Problem 3.7

How many numbers of the form $3n^5 + 11$, when n varies from 1 to 2000, are prime?

$\Longrightarrow$ Solution.

First, let us produce the first 20 numbers of this form.

```
plist=Table[3 n^5 + 11, {n, 1, 20}]
{14, 107, 740, 3083, 9386, 23339, 50432, 98315, 177158,
300011, 483164, 746507, 1113890, 1613483, 2278136, 3145739,
4259582, 5668715, 7428308, 9600011}
```

The next step, in the spirit of Section 3.2, would be to apply `PrimeQ` to all the numbers and find out which ones are prime. Since this is a listable function it is enough:

```
PrimeQ[plist]
{False, True, False, True, False, True, False, False, False,
False, False, True, False, True, False, True, False, False,
False, False}
```

Now we are left to count the number of `True` ones. This is do-able here; 6 of these numbers are prime. However, if we are dealing with a list with 2000 elements, we need to select, from the elements of the list, those with a desired property,[1] here being prime. The command `Select` does just this:

[1] Or a desired pattern, more about this later.

```
Select[plist,PrimeQ]
{107, 3083, 23339, 746507, 1613483, 3145739}
```

These are prime numbers in the list `plist`. The command `Length` gives the length of a list. If we assemble all the steps in one line we have

```
Length[Select[Table[3 n^5 + 11, {n, 1, 20}], PrimeQ]]
6
```

Thus to find out how many numbers of the form $3n^5 + 11$ are prime when n runs from 1 to 2000, all we have to do is to change 20 to 2000:

```
Length[Select[Table[3 n^5 + 11, {n, 1, 2000}], PrimeQ]]
97
```

In a nutshell, `Select[list,f]`, will apply the function `f` (which returns `True` or `False`) to all the elements, say x, of the `list` and return those elements for which `f[x]` is true. A function which returns either `True` or `False` is called a *Boolean* function. So far we have seen one Boolean function, that is, `PrimeQ`. (More on Boolean statements in Chapter 5.)

Problem 3.8

Show that among the first 500 Fibonacci numbers, 18 of them are prime.

$\Longrightarrow$ Solution.

We do this step by step. First we generate the first 10 Fibonacci numbers. Then among those we select the ones which are prime. Then using `Length` we get the size of this list, as follows:

```
fiblist=Table[Fibonacci[i], {i, 1, 10}]
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55}

Select[fiblist, PrimeQ]
 {2, 3, 5, 13}

Length[Select[Table[Fibonacci[i], {i, 1, 10}], PrimeQ]]
4
```

Now instead of 10 we generate 500 numbers:

```
Length[Select[Table[Fibonacci[i], {i, 1, 500}], PrimeQ]]
18
```

Exercise 3.2

Show that, among the first 450 Fibonacci numbers, the number of odd Fibonacci numbers is twice the number of even ones. (Hint: see `OddQ` and `EvenQ` for odd and even numbers.)

Exercise 3.3

Let m be a natural number and

$$A = \frac{(m+3)^3 + 1}{3m}.$$

Find all the integers m less than 500 such that A is an integer. Show that A is always odd. (Hint: see `IntegerQ`.)

Let us look at another example, slightly different but of the same nature as Problem 3.7. The following example shows that anonymous functions fit very well with `Select`.

Problem 3.9

For which $1 \leq n \leq 1000$ does the formula $2^n + 1$ produce a prime number?

$\Longrightarrow$ Solution. Here is the solution:

```
Select[Range[1000], PrimeQ[2^(#) + 1] &]
{1, 2, 4, 8, 16}
```

Let us take a deep breath and go through this one-liner code slowly. The function `PrimeQ[2^(#)+1]&` is an anonymous function which gives `True` if the number $2^n + 1$ is prime and `False` otherwise.

```
PrimeQ[2^(#) + 1] &[12]
False
```

`Range[1000]` creates a list containing the numbers from 1 to 1000. The command `Select` applies the anonymous function `PrimeQ[2^(#)+1]&` to each element of this list and in the cases when the result is true, i.e., when $2^n + 1$ is prime, the element will be selected. Thus {`1, 2, 4, 8, 16`} are the only numbers that make $2^n + 1$ a prime number.

Exercise 3.4

Find the number of positive integers $0 < n < 20000$ such that 1997 divides $n^2 + (n+1)^2$. Try the same code for 2009.

Exercise 3.5

For integers $2 \leq n \leq 200$, find all n such that n divides $(n-1)! + 1$. Show that there are 46 such n.

Problem 3.10

Notice that $12^2 = 144$ and $21^2 = 441$, namely the numbers and their squares are reverses of each other. Find all the numbers up to 10000 with this property.

$\Longrightarrow$ Solution.

We need to introduce some new built-in functions. `IntegerDigits[n]` gives a list of the decimal digits of the integer n. We also need `Reverse` and `FromDigits`:

```
IntegerDigits[80972]
{8, 0, 9, 7, 2}

Reverse[%]
{2, 7, 9, 0, 8}

FromDigits[%]
27908
```

Thus the above shows we can easily produce the reverse of a number:

```
re[n_] := FromDigits[Reverse[IntegerDigits[n]]]

re[12345]
54321

re[6548629]
9268456
```

Having this function under our belt, the solution to the problem is just one line. Notice that the problem is asking for the numbers n such that re[n^2]=re[n]^2 .

```
Select[Range[10000], re[#]^2 == re[#^2] &]
{1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 30, 31, 100, 101,
102, 103, 110, 111, 112, 113, 120, 121, 122, 130, 200, 201, 202,
210, 211, 212, 220, 221, 300, 301, 310, 311, 1000, 1001, 1002,
1003, 1010, 1011, 1012, 1013, 1020, 1021, 1022, 1030, 1031, 1100,
1101, 1102, 1103, 1110, 1111, 1112, 1113, 1120, 1121, 1122, 1130,
1200, 1201, 1202, 1210, 1211, 1212, 1220, 1300, 1301, 2000, 2001,
2002, 2010, 2011, 2012, 2020, 2021, 2022, 2100, 2101, 2102, 2110,
2111, 2120, 2121, 2200, 2201, 2202, 2210, 2211, 3000, 3001, 3010,
3011, 3100, 3101, 3110, 3111, 10000}
```

Here is one more example using the command `FromDigits`. We know that 11 is a prime number. One wonders what is the next prime number consisting only of ones. A wild guess: a number with 23 digits all one? All we have to do is to produce this number then, with `PrimeQ`, test whether this is prime. Here is one way to generate this number.

```
Table[1, {23}]
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1}

FromDigits[%]
11111111111111111111111

PrimeQ[%]
True
```

Here is the code to find out which numbers of this kind up to 500 digits are prime.

```
Select[Range[500], PrimeQ[FromDigits[Table[1, {#}]]] &]
{2, 19, 23, 317}
```

Problem 3.11

Show that the number of k between 0 and 1000 for which $\binom{1000}{k}$ is odd is a power of 2.

$\Longrightarrow$ Solution.

First, `Range[0,1000]` creates a list of numbers from 0 to 1000. Using an anonymous function, each time we plug these numbers into `Binomial[1000, #]` and check right away if this is an odd number by `OddQ[Binomial[1000, #]] &`. If this is the case, `Select` will pick up these numbers. The rest is clear from the code below.

```
Select[Range[0, 1000], OddQ[Binomial[1000, #]] &]
{0, 8, 32, 40, 64, 72, 96, 104, 128, 136, 160, 168, 192,
200, 224, 232, 256, 264, 288, 296, 320, 328, 352, 360, 384, 392,
416, 424, 448, 456, 480, 488, 512, 520, 544, 552, 576, 584, 608,
616, 640, 648, 672, 680, 704, 712, 736, 744, 768, 776, 800, 808,
832, 840, 864, 872, 896, 904, 928, 936, 960, 968, 992, 1000}

Length[Select[Range[0, 1000], OddQ[Binomial[1000, #]] &]]
64

FactorInteger[%]
{{2, 6}}
```

Exercise 3.6

Show that among the first 200 primes p, the ones such that the remainder when 19^{p-1} is divided by p^2 is 1 are $\{3, 7, 13, 43, 137\}$.

Exercise 3.7

An integer $d_n d_{n-1} \dots d_1$ is called *prime-palindromic* if

$$d_n d_{n-1} \dots d_1 \text{ and } d_1 \dots d_{n-1} d_n$$

are both prime (for example 941). Write a code to find all prime-palindromic numbers up to 5000. Observe that there are 167 such numbers.

Problem 3.12

Find all positive integers $0 < m < 10^5$ such that the fourth power of the number of positive divisors of m equals m. (Hint: see `Divisors`.)

$\Longrightarrow$ Solution.

The function `Divisors[n]` gives all the positive numbers which divide n. This is a list which includes 1 and n also. We need to get the number of these divisors, thus the function `Length`. We are looking for each number m such that the fourth power of `Length[Divisors[m]]` is `m`. The rest is clear from the code.

```
Select[Range[10^5], Length[Divisors[#]]^4 == # &]
{1, 625, 6561}
```

Exercise 3.8

Show that there is only one positive integer n smaller than 1000 such that $n! + (n+1)!$ is the square of an integer.

The idea of sending a function into a list, i.e., applying a function to each element of a list, seems to be a good one. We have already mentioned that the listable built-in functions are able to go inside a list, like `PrimeQ` or `Prime`. Have a look at the Attributes of `Prime` in the following:

```
??Prime
Prime[n] gives the nth prime number.
Attributes[Prime] = {Listable, Protected}
```

Also recall that if a function is not listable, `Map` or `/@` will push the function into the list as the following problem demonstrates:

Problem 3.13

What digit does not appear as the last digit of the first 20 Fibonacci numbers?

$\Longrightarrow$ Solution.

This one-liner code collects all the digits which appears as the last digit:

```
Union[Last /@ (IntegerDigits /@ (Fibonacci /@ Range[20]))]
{0, 1, 2, 3, 4, 5, 7, 8, 9}
```

Thus 6 is the only digit which is not present. Let us understand this code. As the command `/@` applies a function to all elements of a list, `Fibonacci /@ Range[20]` produces the first 20 Fibonacci numbers.

```
Fibonacci /@ Range[20]
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610,
987, 1597, 2584, 4181, 6765}
```

Then `IntegerDigits` would go inside this list and get the digits of each number.

```
IntegerDigits/@ Fibonacci /@ Range[20]
{{1}, {1}, {2}, {3}, {5}, {8}, {1, 3}, {2, 1}, {3, 4}, {5, 5}, {8,
  9}, {1, 4, 4}, {2, 3, 3}, {3, 7, 7}, {6, 1, 0}, {9, 8, 7},
  {1, 5, 9, 7}, {2, 5, 8, 4}, {4, 1, 8, 1}, {6, 7, 6, 5}}
```

Then the function `Last` will get the last digits as required.

```
Last /@ IntegerDigits /@ Fibonacci /@ Range[20]
{1, 1, 2, 3, 5, 8, 3, 1, 4, 5, 9, 4, 3, 7, 0, 7, 7, 4, 1, 5}
```

`Union` will get rid of any repetitions in the list. (More on this command in Section 5.2.)

Since `Fibonacci` and `IntegerDigits` are listable functions, one can also write the above code as follows:

```
Union[Last /@ IntegerDigits[Fibonacci[Range[20]]]]
```

If one does not want to use `IntegerDigits` then one can use the `Mod` function to get access to the last digit of a number.

```
?Mod
Mod[m, n] gives the remainder on division of m by n.

Mod[264,10]
4
?Quotient
Quotient[m, n] gives the integer quotient of m and n.

Quotient[264,10]
26

26*10+4
264
```

Thus another way to write the code is as follows. Note that `Mod` is also a listable function.

```
Union[Mod[Fibonacci[Range[20]],10]]
{0, 1, 2, 3, 4, 5, 7, 8, 9}
```

In Section 5.2 we will see how to use *Mathematica* to get directly the digit 6 that we are after, namely to handle sets.

Recall that one can decompose any number n as a product of powers of primes and this decomposition is unique, i.e., $n = p_1^{k_1} \cdots p_t^{k_t}$ where p_i's are prime. Let us call a number a *square free* number if, in its decomposition to primes, all the k_i's are 1. Namely, no power of primes can divide this number. Thus $15 = 3 \times 5$ is a square free number but $48 = 2^4 \times 3$ is not.

Recall that `Select[list,f]` will apply the function `f` (which returns `True` or `False`) to all the elements, say x, of the `list` and return those elements for which `f[x]` is true. There is an option in `Select` which makes it possible to get only the first n elements of `list` that satisfy `f`, i.e., `f` returns `True`, as follows: `Select[list,f,n]`. This comes in very handy, as in some problems we are only interested in a certain number of data which satisfy `f`. Also this can be used in circumstances where we want to test the elements until something goes wrong or some desirable element comes up. The following example demonstrates this.

Problem 3.14

Write a function `squareFreeQ[n]` that returns `True` if the number `n` is a square free number, and `False` otherwise.

$\Longrightarrow$ Solution.

Here is the code:

```
squareFreeQ[n_]:= Select[Last/@ FactorInteger[n], # != 1&,1]
== {}

squareFreeQ[2*5*7]
True

squareFreeQ[2*5*7*7]
False
```

Let us decipher this code. Recall from Section 1.2 that if $n = p_1^{k_1} \cdots p_t^{k_t}$ is the decomposition of n into its prime factors then

```
FactorInteger[n]
```

$$\{\{p_1, k_1\}, \{p_2, k_2\}, \cdots, \{p_t, k_t\}\}$$

Now all we have to do is to go through this list and see if all k_i's are one. So the first step is to apply `Last` to each list to discard p_i's and get k_i's.

```
Last /@ FactorInteger[n]
```

$$\{k_1, k_2, \cdots, k_t\}$$

Having this list, we shall go through the list one by one and examine if k_i's are one. The anonymous function `#≠ 1&` does exactly this. So `Select[`$\{k_1, k_2, \cdots, k_t\}$`,#≠ 1&]` gives the list of k_i's which are not one. But in our case, looking for square free primes, it is enough if only one k_i is not one. Then the number is not square free. Thus we use an option of `Select` which goes through the list until it finds an element such that k_i is not one. So we need to modify the code to `Select[`$\{k_1, k_2, \cdots, k_t\}$`,#≠ 1&,1]`. We are almost done. All we have to do is to see if this list is empty or not (namely is there any k_i not equal to one). And for this we compare `Select[`$\{k_1, k_2, \cdots, k_t\}$`,#≠ 1&,1]=={}`.

We can solve this problem later with a slightly different method (see Problem 4.3).

Exercise 3.9

Find the first 5 positive integers n such that $n^6 + 1091$ is prime. Show that all these n are between 3500 and 8500.

Exercise 3.10

A number with n digits is called *cyclic* if multiplication by $1, 2, 3, \cdots, n$ produces the same digits in a different order. Find the only 6-digit cyclic number.

Problem 3.15

Find out how many primes bigger than n and smaller than $2n$ exist, when n goes from 1 to 30.

$\Longrightarrow$ Solution.

We define an anonymous function which finds all the primes bigger than n and smaller than $2n$ and then gets the size of this list. Once we are done with this, we apply this function to a list of numbers from 1 to 30. Our anonymous function looks like this:

```
Length[Select[Range[# + 1, 2 # - 1], PrimeQ]] &
```

Here, `Range[# + 1, 2 # - 1]` produces all the numbers between n and $2n$. Then `Select` finds out which of them are in fact prime. Then we use the command `Length` to get the number of elements of this list. One can optimize this a bit, as we don't need to look at the whole range of n to $2n$, as clearly even numbers are not prime so we can ignore them right from the beginning. But we leave it to the reader to do this. All we have to do now is to apply this function with `Map` or `/@` to numbers from 1 to 30.

```
Length[Select[Range[# + 1, 2 # - 1], PrimeQ]] & /@ Range[30]
{0, 1, 1, 2, 1, 2, 2, 2, 3, 4, 3, 4, 3, 3, 4, 5, 4, 4, 4, 4, 5,
6, 5, 6, 6,  6, 7, 7, 6, 7}
```

One can also write a more innocent code, using `Table` as follows:

```
 Table[Length[Select[Range[n + 1, 2n - 1], PrimeQ]], {n, 1, 30}]
{0, 1, 1, 2, 1, 2, 2, 2, 3, 4, 3, 4, 3, 3, 4, 5, 4, 4, 4, 4,
5, 6, 5, 6, 6, 6, 7, 7, 6, 7}
```

This doesn't seem to be the most efficient way to write this problem as each time we test the same numbers repeatedly to see whether they are prime. We will write another code for this problem in Problem 7.1 using a `Do` Loop. One can also use the built-in function `PrimePi` to write a very short solution for this problem. For the usage of `PrimePi` see Problem 1.3.

A word is called *palindromic* if it reads the same backwards as forwards, e.g., madam. In the next problem we are going to find all the palindromic words in a lot of languages! The approach is similar to Problem 3.10, however, in Problem 3.16 we will be working with strings of characters.

Problem 3.16

Write a function to check whether a word is palindromic.
Using `DictionaryLookup`, find all the palindromic words in the English language. Then draw a bar chart showing how many palindromic words there are in different languages which are supported by *Mathematica*.

$\Longrightarrow$ Solution.

We first define a function to check whether a word is palindromic. Using `StringReverse`, this is easy:

```
pal[n_] := n == StringReverse[n]

pal["test"]
False

pal["kayak"]
True
```

Now we select from the English dictionary available in *Mathematica* all the words which are palindromic.

```
Select[DictionaryLookup[], pal]
{"a", "aha", "aka", "bib", "bob", "boob", "bub", "CFC",
"civic", "dad", "deed", "deified", "did", "dud", "DVD", "eke",
"ere", "eve", "ewe", "eye", "gag", "gig", "huh", "I", "kayak",
"kook", "level", "ma'am", "madam", "mam", "MGM", "minim",
"mom", "mum", "nan", "non", "noon", "nun", "oho", "pap",
"peep", "pep", "pip", "poop", "pop", "pup", "radar", "redder",
"refer", "repaper", "reviver", "rotor", "sagas", "sees", "seres",
"sexes", "shahs", "sis", "solos", "SOS", "stats", "stets", "tat",
"tenet", "TNT", "toot", "tot", "tut", "wow", "WWW"}

Length[Select[DictionaryLookup[], pal]]
70
```

This shows there are 70 palindromic words in this dictionary. We will do the same with other languages, but first let us see what languages are available.

```
DictionaryLookup[All]
{"Arabic", "BrazilianPortuguese", "Breton",
"BritishEnglish", "Catalan", "Croatian", "Danish", "Dutch",
"English", "Esperanto", "Faroese", "Finnish", "French",
"Galician", "German", "Hebrew", "Hindi", "Hungarian",
"IrishGaelic", "Italian", "Latin", "Polish", "Portuguese",
"Russian", "ScottishGaelic", "Spanish", "Swedish"}
```

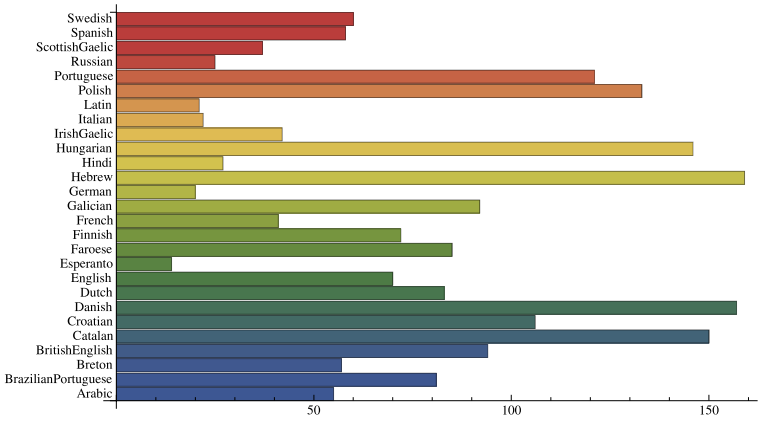
We will choose all the palindromic words from all these dictionaries. The following is going to take about a minute.

```
t = Table[{lan, Length[Select[DictionaryLookup[{lan, All}],
pal]]},   {lan,DictionaryLookup[All]}]

{{"Arabic", 55}, {"BrazilianPortuguese", 81}, {"Breton",
  57}, {"BritishEnglish", 94}, {"Catalan", 150}, {"Croatian",
  106}, {"Danish", 157}, {"Dutch", 83}, {"English", 70},
  {"Esperanto", 14}, {"Faroese", 85}, {"Finnish", 72},
  {"French", 41}, {"Galician", 92}, {"German", 20},
  {"Hebrew", 159}, {"Hindi", 27}, {"Hungarian",
  146}, {"IrishGaelic", 42}, {"Italian", 22}, {"Latin",
  21}, {"Polish", 133}, {"Portuguese", 121}, {"Russian",
  25}, {"ScottishGaelic", 37}, {"Spanish", 58}, {"Swedish", 60}}
```

We are now ready to put these on a Bar chart.

```
palnum = Last /@ t
{55, 81, 57, 94, 150, 106, 157, 83, 70, 14, 85, 72, 41, 92,
20, 159, 27, 146, 42, 22, 21, 133, 121, 25, 37, 58, 60}

BarChart[palnum, BarOrigin -> Left,
 ChartStyle -> "DarkRainbow", ChartLabels ->
 DictionaryLookup[All]]
```

Out[160]=



♣ TIPS

– In Problem 3.16 we used the command `StringReverse`, which reverses the order of the characters, so

```
StringReverse["this is great"]
"taerg si siht"
```

Mathematica has several more commands to work with strings, including `StringLength`, `StringTake`, `StringDrop`, `StringReplace`, `ToString`, etc.

Problem 3.17

Find all the words in the *Mathematica* dictionary such that the reverses of the words are in the dictionary as well (for example, loots and stool).

⟹ Solution.

There are 496 words in the *Mathematica* dictionary such that their reverses have a meaning as well. Here is the code:

```
Select[DictionaryLookup[], {StringReverse[#]} ==
   DictionaryLookup[StringReverse[#] ] &]
```

However, as we don't want to print all the words, we use `Short` to get one line of the list.

```
?Short

Short[expr] prints as a short form of expr, less than about
one line long.
Short[expr,n] prints as a form of expr about n lines long. >>

Short[
 Select[DictionaryLookup[], {StringReverse[#]} ==
    DictionaryLookup[StringReverse[#] ] &]]

{a,abut,agar,ah,aha,<<486>>,yaps,yard,yaw,yaws,yob}
```

4 Changing heads!

Mathematica considers expressions in a unified manner. Any expression consists of a head and its arguments. For example {a,b,c} is considered as `List[a,b,c]` with the function `List` as the head and `a,b,c` as its arguments. *Mathematica* enables us to replace the head of an expression with another expression. For example, replacing the head of {a,b,c} with `Plus` gives `Plus[a,b,c]` which is `a+b+c`. This simple idea provides a powerful method to approach solving problems as this chapter demonstrates.

Let us for a moment be a bit abstract. Wolfram *Mathematica*® has a very consistent way of dealing with expressions. Any expression in *Mathematica* has the following presentation `head[arg1,arg2,...,argn]` where `head` and `arg` could be expressions themselves. To make this point clear let us use the command `FullForm` which shows how *Mathematica* considers an expression.

```
FullForm[a + b + c]
Plus[a, b, c]

FullForm[a*b*c]
Times[a, b, c]

FullForm[{a, b, c}]
List[a, b, c]
```

Notice that the expressions `a+b+c` and {a,b,c} which represent very different things have such close presentations. Here `Plus` is a function and `a,b,c` are plugged into this function. `Plus` is the head of the expression `a+b+c`. One can see from the `FullForm` that the only difference between `a+b+c` and {a,b,c} is their heads! We can get the head of any expression:

```
Head[{a, b, c}]
List

Head[a + b + c]
Plus

{a, b, c}[[0]]
List
```

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_4, © Springer-Verlag London Limited 2010

Mathematica gives us the ability to replace the head of an expression with another head. The consequence of this is simply (head and) mind-blowing!

This can be done with the command `Apply`. Here is the traditional example:

```
Apply[Plus,{a,b,c}]
a+b+c
```

It is not difficult to explain this. The full form of {a,b,c} is `List[a,b,c]` with the head `List`. All *Mathematica* does is to change the head `List` to `Plus`, thus we have `Plus[a,b,c]` which is `a+b+c`.

The shorthand for `Apply` is `@@`, as the following example shows:

```
Plus @@ Range[10]
55
```

This gives the sum of 1 to 10.

Problem 4.1

Define the following functions:

$$\begin{aligned} cs(n) &= 1^3 + 2^3 + \cdots + n^3 \\ ep(n) &= 1 + \frac{1}{1} + \frac{1}{2!} + \cdots + \frac{1}{n!} \\ p(n) &= (1+x)(1+x^2)\cdots(1+x^n). \end{aligned}$$

$\Longrightarrow$ Solution.

Recall that `Range[n]` gives the list $\{1, 2, ..., n\}$ and since `^` is a listable function, `Range[n]^3` produces $\{1^3, 2^3, ..., n^3\}$. Therefore, as in the example above, if we replace the head of the list with `Plus`, i.e., `Plus @@ Range[n]^3` we will have $1^3 + 2^3 + ... + n^3$. The other functions use a similar approach.

```
cs[n_]:=Plus @@ Range[n]^3

ep[n_]:=1.+Plus @@ (1/Range[n]!)

p[n_]:=Times @@ (1+x^Range[n])
```

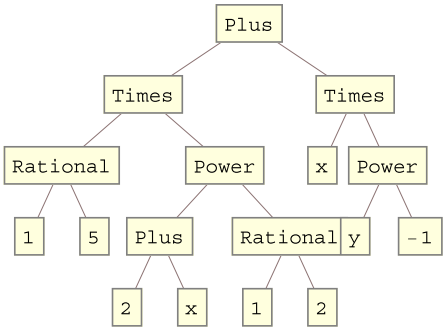
Let us look at the second example also. Again, `Range[n]` produces a list $\{1, 2, 3, \cdots, n\}$. Note that the factorial function `!` is listable, thus `Range[n]!` would produce $\{1!, 2!, 3!, \cdots, n!\}$. Recall that all the arithmetic operations are also listable, including `/` . Thus `1/Range[n]!` produces $\{\frac{1}{1!}, \frac{1}{2!}, \frac{1}{3!}, \cdots, \frac{1}{n!}\}$. We are almost there, all we have to do is to replace the head of $\{\frac{1}{1!}, \frac{1}{2!}, \frac{1}{3!}, \cdots, \frac{1}{n!}\}$ which is a `List` with `Plus` and as a result we get $\frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!}$.

All these functions are classical examples of using `Sum` and `Product` which are available in *Mathematica*. We will meet these commands in Chapter 6.

As the expressions get more complicated, it is quite difficult to analyze the `FullForm` of an expression (if this is necessary at all). The command `TreeForm` is an excellent facility to shed more light on how the functions are composed to get the expression. Here is an example:

```
FullForm[Sqrt[2 + x]/5 + x/y]

Plus[Times[Rational[1,5], Power[Plus[2,x], Rational[1,2]]],
Times[x, Power[y, -1]]]
```

```
TreeForm[Sqrt[2 + x] / 5 + x / y]
```



To understand this, start from the bottom of the tree and make your way to the top and you will get the expression

$$\frac{\sqrt{2+x}}{5}+\frac{x}{y}.$$

Problem 4.2

Show that the only n less than 1000 such that

$$3^n+4^n+\cdots+(n+2)^n=(n+3)^n$$

are the numbers 2 and 3.

$\Longrightarrow$ Solution.

We first need to write a code to evaluate $3^n+4^n+\cdots+(n+2)^n$. This is very similar to the functions in Problem 4.1 and it looks like this

```
Apply[Plus, Range[3, n+2]^n]
```

Or, if using the shorthand, the equivalent code is `Plus @@ Range[3, n + 2]^ n`. We then select from the range of 1 to 1000, those in which the result of the above is the same as `(n+3)^ n`. Note that in order to plug these into `Select` we need to make these two into one anonymous function:

```
Select[Range[1000],Apply[Plus, Range[3, # + 2]^#] == (# + 3)^# &]
{2, 3}
```

Exercise 4.1

A number is called a *Harshad* number[1] if it is divisible by the sum of its digits (e.g., 12 is Harshad as it is divisible by 1+2=3). Find all 2-digit Harshad numbers. How many 5-digit Harshad numbers are there?

Let's look at Problem 3.14 again.

Problem 4.3

Write a function `squareFreeQ[n]` that returns `True` if the number n is a square free number, and `False` otherwise.

$\Longrightarrow$ Solution.

Here is the whole code:

```
squareFree1Q[n_] := Times @@ Last /@ FactorInteger[n] == 1

squareFree1Q /@ {12, 13, 14, 25, 26}
{False, True, True, False, True}
```

It might take a while to understand how this code works. If $n = p_1^{k_1} \cdots p_t^{k_t}$, then `FactorInteger[n]` will produce $\{\{p_1, k_1\}, \{p_2, k_2\}, \cdots, \{p_t, k_t\}\}$. We are after numbers such that all the k_i are 1 in the decomposition. Thus we can get all the k_i, multiply them and if we get anything other than 1, then this would be a non-square free number. Thus the first step is to apply `Last` to the list `Last /@ FactorInteger[n]` to get $\{k_1, k_2, \cdots, k_t\}$. Then all we have to do is to multiply them all together, and here comes the `Times @@` to change the head of $\{k_1, k_2, \cdots, k_t\}$ from `List` to `Times`.

[1] Harshad means "giving joy" in Sanskrit, named and defined by the Indian mathematician D. Kaprekar.

Problem 4.4

Find all the numbers up to one million which have the following property: if $n = d_1 d_2 \cdots d_k$ then $n = d_1! + d_2! + \cdots + d_k!$ (e.g. $145 = 1! + 4! + 5!$).

$\Longrightarrow$ Solution.

```
Select[Range[1000000], Plus @@ Factorial /@ IntegerDigits[#]
== # &]
{1, 2, 145, 40585}
```

The code consists of an anonymous function which, for any n, checks whether it has the desired property of the problem. Then, by using `Select`, we check the list of all the numbers from 1 to one million, `Range[1000000]`. Our anonymous function is `Plus @@ Factorial /@ IntegerDigits[#] == # &`. Let us look at the left-hand side of `==`. The built-in function `IntegerDigits[#]` applied to $n = d_1 d_2 \cdots d_k$ produces the list of digits of n, namely $\{d_1, d_2, \cdots, d_k\}$. Next, applying `Factorial /@` to this list, we get $\{d_1!, d_2!, \cdots, d_k!\}$. Now all we need is to get the sum of elements of this list, and this is possible by changing the head from `List` to `Plus` by `Plus @@`. Once this is done, we compare the left-hand side of `==` with the right-hand side which is the original number `#`.

Problem 4.5

A number is *perfect* if it is equal to the sum of its proper divisors, e.g., $6 = 1 + 2 + 3$ but $18 \neq 1 + 2 + 3 + 6 + 9$. Write a program to find all the perfect numbers up to 10000. (Hint, have a look at the command `Divisors`.)

$\Longrightarrow$ Solution.

Here is a step-by-step approach to the solution.

```
Divisors[6]
{1,2,3,6}

Most[Divisors[6]]
{1,2,3}

Apply[Plus,Most[Divisors[6]]]
6

Select[Range[10000], # == Apply[Plus, Most[Divisors[#]]] &]
{6, 28, 496, 8128}
```

The numbers 6, 28 and 496 were already known as being perfect numbers 2000 years before Christ. A glance at the list shows that all the perfect num-

bers we have found are even. It is still unknown whether there is an odd perfect number. Probably this is the oldest unsolved question in mathematics!

Problem 4.6

Among the first 100000 numbers, what is the largest number n which is divisible by all positive integers $\leq \sqrt{n}$?

$\Longrightarrow$ SOLUTION.

```
Select[Range[100000],(Mod[#,LCM @@ Range[Floor[Sqrt[#]]]]==0)&]
{1, 2, 3, 4, 6, 8, 12, 24}
```

Exercise 4.2

Decipher what the following codes do:

```
g[n_] := Times @@ Apply[Plus , Inner[List, x^Range[n],
1/x^Range[n], List], 1]

t[n_] := Times @@ Apply[Plus, Thread[List[x^Range[n],
1/x^Range[n]]], 1]
```

Exercise 4.3

Find all the numbers up to 100 which have the following property: if $n = p_1^{k_1} \cdots p_t^{k_t}$ is the prime decomposition of n then $n = k_1 \times p_1 + k_2 \times p_2 + \cdots + k_t \times p_t$.

Exercise 4.4

Consider all numbers of the form $3n^2+n+1$, where $0 < n < 10000$. How small can the sum of the digits of such a number be? (Hint, see `Min`.)

Exercise 4.5

Decipher what the following codes do:

```
Power @@ (x + y)

Plus @@@ (x^y + y^z)
```

Exercise 4.6

Show that the sum of all the divisors of the number

$$6086555670238378989670371734243169622657830773351885970528324860512791691264$$

is a perfect number. (This number is the only known sublime number besides 12. A *sublime number* is a positive integer which has a perfect number of positive divisors (including itself), and whose positive divisors add up to another perfect number. See Problem 4.5 for perfect numbers and MathWorld [7] for sublime numbers.)

Exercise 4.7

A *weird number* is a number such that the sum of the proper divisors (divisors including 1 but not itself) of the number is greater than the number, but no subset of those divisors sums to the number itself. Find all the weird numbers up to 10000.

5
A bit of logic and set theory

This chapter focuses on statements which take values of `True` or `False`, i.e., Boolean statements. We can combine statements of this type to make decisions based on their values. We introduce the `If` statement available in *Mathematica*.

5.1 Being logical

In mathematical logic, statements can have a value of `True, False` or undefined.[1] These are called *Boolean expressions*. This helps us to "make a decision" and write programs based on the value of a statement (I am thinking of the classical `If-Then` statement; `If` something is True, `Then` do this, `Otherwise` do that). We have seen `==` which compares the left-hand side with the right-hand side. Studying the following examples carefully will tell us how Wolfram *Mathematica*® approaches logical statements:

```
3^2+4^2==5^2
True

3^2+4^2>5^2
False

9Sqrt[10!] < 10Sqrt[9!]
False

(x-1)(x+1)==x^2-1
(x-1)(x+1)==x^2-1
```

[1] We don't want to go into detail here mainly because I don't know the detail!

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_5, © Springer-Verlag London Limited 2010

```
Simplify[%]
True

x==5
x==5

{1,2}=={2,1}
False

{a,b}=={b,a}
{a,b}=={b,a}
```

As one notices, *Mathematica* echoes back the expressions that she can't evaluate (e.g., x==5). Among them `{a,b}=={b,a}`, although one expects to get `False` as lists respect order. This is because *Mathematica* does not know about the values of a and b, and in case a and b are the same then `{a,b}=={b,a}` is `True`, and `False` otherwise. If you want *Mathematica* to judge from the face value, then use ===,

```
x==5
False

{a,b}=={b,a}
False

?===
ihs===rhs yields True if the expression lhs is identical
to rhs and yields False otherwise. >>
```

One can combine logical statements with usual operations `And, Or, Not,` or the equivalent &&, ||, ! as the following examples show:

```
2 > 3 && 3 > 2
False

And[2 > 3, 3 > 2]
False

1 < 2 < 3
True

2 > 3 || 2 < 3
True

Or[2 > 3, 3 > 2]
True

3^2+4^2>= 5^2
True
```

In general, for two statements $\mathcal{A}$ and $\mathcal{B}$, the statement $\mathcal{A}\&\&\mathcal{B}$ is false if one of $\mathcal{A}$ or $\mathcal{B}$ is false and $\mathcal{A}$ || $\mathcal{B}$ is true if one of them is true. In order to produce all possible combinations of true and false cases, we use the command `Outer`

as the following example shows (we will look at the command `Outer` in more detail in Section 7.5).

```
Outer[f, {a, b}, {x, y}]
{{f[a, x], f[a, y]}, {f[b, x], f[b, y]}}
```

Thus, if in the above we replace `f` by `And` or `Or` we will get all the possible combinations of `True` and `False`.

```
Outer[And, {True, False}, {True, False}]
{{True, False},{False,False}}

Outer[Or, {True, False}, {True, False}]
{{True, True}, {True,False}}
```

In *Mathematica*, for a variable, one can specify certain domains. This means that the variable takes its values from a specific type of data. The domains available are `Algebraics, Booleans, Complexes, Integers, Primes, Rationals` and `Reals`. One of the fundamental theorems in number theory is to show that π is not a rational number, i.e., is not of the form m/n, where m and n are integers. Look at the following examples:

```
Pi ∈ Rationals
False

Sqrt[7] ∈ Integers
False
```

Problem 5.1

Show that $1+\sqrt{3}+\sqrt{5}+\sqrt{7}$ is an algebraic number (i.e., it is a solution of a polynomial equation with integer coefficients).

$\Longrightarrow$ Solution.

```
Plus @@ Sqrt[Range[1, 7, 2]] ∈ Algebraics
True
```

The last example shows that $1+\sqrt{3}+\sqrt{5}+\sqrt{7}$ is an algebraic number (i.e. it is a solution of a polynomial equation with integer coefficients).

One can use membership ($\in$) to write neat solutions to several problems.

Problem 5.2

Does the formula $(n!)^2+1$ give prime numbers for $n=1$ to 6?

$\Longrightarrow$ Solution.

```
(#!^2 + 1) & /@ Range[6] ∈ Primes
False
```

Here we first apply the anonymous function `(#!^2 + 1)` which is the formula $(n!)^2 + 1$ to the list containing 1 to 6. Then we ask *Mathematica* if the elements of this list belong to the domain `Primes`. The answer is `False`. The following code shows that the above formula does not produce a prime number for $n = 6$:

```
PrimeQ /@ ((#!^2 + 1) & /@ Range[1, 6])
{True, True, True, True, True, False}
```

One should be careful that *Mathematica* cannot (yet) perform miracles. For example, one can actually prove that $\sqrt[3]{2+\sqrt{5}} + \sqrt[3]{2-\sqrt{5}}$ is an integer, but

```
(2 + 5^(1/2))^(1/3) + (2 - 5^(1/2))^(1/3) ∈
Integers
False

FullSimplify[(2 + 5^(1/2))^(1/3) +
        (2 - 5^(1/2))^(1/3) ∈ Integers]
False
```

Mathematica provides the logical quantifiers $\forall$, $\exists$ and $\Rightarrow$ with `ForAll`, `Exist` and `Implies` commands. But these seem not to be that powerful yet. For example, one cannot prove Fermat's little theorem which says $2^{p-1} \equiv 1 (\mathrm{mod}\ p)$ where $p > 2$ is a prime number with them!

```
ForAll[p, p ∈ Primes, Mod[2^(p - 1), p] == 1]
```

Or even an easy fact that the product of four consecutive numbers plus one is a squared number.

```
Implies[n ∈ Integers && n > 0, Sqrt[n(n + 1)(n +
2)(n + 3) + 1] ∈ Integers]
```

In both cases *Mathematica* gives back the same expression, indicating she cannot decide on them.

5.2 Handling sets

Now it has been agreed among mathematicians that *any* mathematics starts by considering sets, i.e., collections of objects.[2] As we mentioned, the difference between mathematical sets and lists in *Mathematica* is that lists respect order and repetition, which is to say one can have several copies of one object in a list (see Chapter 3). Sets are not sensitive about repeated objects, e.g., the set $\{a, b\}$ is the same as the set $\{a, b, b, a\}$. There is no concept of sets in *Mathematica* and if necessary one considers a list as a set.

If one wants to get rid of duplications in a list, one can use

```
Union[{a,b,b,a}]
{a,b}
```

Considering two sets, the natural operations between them are union and intersection. *Mathematica* provides `Union` to collect all elements from different lists in one list (after removing all the duplications) and `Intersection` for collecting common elements (again discarding repeated elements). The following examples show how these commands work.

```
u= {1, 2, 3, 4, 5, 2, 4, 7, 4}; a = {1, 4, 7, 3}; b
= {5, 4, 3, 2};

Union[u]
{1, 2, 3, 4, 5, 7}

a ∪ b

{1, 2, 3, 4, 5, 7}

a ∩ b

{3,4}

Complement[u, a]
{2, 5}

?Complement
Complement[eall, e1, e2, ... ] gives the elements in
eall which are not in any of the ei.
```

[2] To be precise, one first considers classes.

```
Complement[u, a ∩ b] == Complement[u, a] ∪
Complement[u, b]
True
```

The first example shows `Union[list]` will get rid of repetition in a list. The command `Complement[u,a]` will give the elements of `u` which are not in `a`. From the example one can see that `a ∩ b` is acceptable in *Mathematica* and is a shorthand for `Intersection[a,b]`. In the last example we checked a theorem of set theory that $(A \cap B)^c = A^c \cup B^c$ where ^{c} stands for complement.

Example 5.3

The following trick will be used later (in Chapter 7, inside a loop) to collect data.

```
A={}
A=A ⋃ {x}
{x}
A=A ⋃ {y}
{x,y}
A=A ⋃ {z}
{x,y,z}
```

This is similar to the traditional trick `sum=sum+i`. Each time `sum=sum+i` is performed, `i` will be added to `sum` and this result will be the new value of `sum`.

There are other ways to add an element to a list.

```
Append[{a,b,c},d]
{a,b,c,d}

A={};A=Append[A,x]
{x}

A=Append[A,y]
{x,y}

A=Append[A,z]
{x,y,z}
```

Also note that the command `AppendTo[s, elem]` is equivalent to `s = Append[s, elem]`.

Problem 5.4

How many positive integers n are there such that n is a divisor of at least one of the numbers 10^{40} and 20^{30}?

$\Longrightarrow$ Solution.

Recall that `Divisors[n]` will produce a list of all the positive integers which divide `n`. So all we need to do is to get the collection of divisors of 10^{40} and 20^{30}. This can be done with `Union`. Note that we are not going to print out all the divisors as this is a long list.

```
Length[Union[Divisors[10^40], Divisors[20^30]]]
2301
```

Problem 5.5

Find out how many common words there are between the English language with French, Dutch and German respectively.

$\Longrightarrow$ Solution.

Recall that `DictionaryLookup` contains all the words in 26 languages (see Problem 3.16). All we have to do is to intersect the list of words in the English language with the list of French words and so on as follows:

```
Length[Intersection[DictionaryLookup[{"English", All}],
  DictionaryLookup[{"French", All}]]]
6897

Length[Intersection[DictionaryLookup[{"English", All}],
  DictionaryLookup[{"Dutch", All}]]]
6166

Length[
 Intersection[DictionaryLookup[{"English", All}],
  DictionaryLookup[{"German", All}]]]
1286
```

♣ TIPS

– The command `Tally[list]` tallies the elements in `list`, listing all distinct elements together with their multiplicities.

– The command `Join[list1,list2]` concatenates lists or other expressions that share the same head.

5.3 Decision making, If and Which

The statement `If[stat,this,that]` where `stat` is a Boolean expression, i.e., has the value of `True` or `False`, will execute `this` if the `stat` value is `True` and `that` otherwise. That means, in either case, one of the statements `this` or `that` will be performed (but not both). So this gives an ability to make a decision about which part one wants to perform. Here is an example:

```
If[12^13+13>13^12+12,
Print["12^13+13>13^12+12"],
Print["12^13+13<13^12+12"]
]
12^13+13>13^12+12
```

This shows that $12^{13} + 13 > 13^{12} + 12$. However, the reader should be cautious, since if it happened that $12^{13} + 13 = 13^{12} + 12$, still the output would have been $12^{13} + 13 < 13^{12} + 12$ (why?). In such situations where there might be more possibilities, the command `Which` suits better. Study this example

```
Which[
 12^13 + 13 > 13^12 + 12, Print["12^13+13>13^12+12"],
 12^13 + 13 < 13^12 + 12, Print["12^13+13<13^12+12"],
 12^13 + 13 == 13^12 + 12, Print["12^13+13=13^12+12"]
 ]
12^13+13>13^12+12
```

Using `If` or `Which`, we are now able to define functions which have conditions.

Problem 5.6

Define the Collatz function as follows:

$$f(x) = \begin{cases} x/2 & \text{if } x \text{ is even} \\ 3x+1 & \text{if } x \text{ is odd.} \end{cases}$$

It was conjectured by L. Collatz in 1937 that if one applies f repeatedly to any number, one eventually arrives at 1. Find out how many times one needs to apply f to 16 in order to reach 1.

$\Longrightarrow$ Solution.

Recall that `EvenQ` is a Boolean statement which returns `True` if the number is even and `False` otherwise. The function f consists of two parts: If n is even, then $f(n) = x/2$; otherwise $f(n) = 3n + 1$. One can use an `If` statement to define this function as follows:

```
f[n_] := If[EvenQ[n], n/2, 3 n + 1]

f[16]
8

f[8]
4

f[4]
2

f[2]
1
```

We will return to this conjecture and will write this function again in Problems 9.6 and 10.2 using the *Mathematica* rules and functions with multiple definitions, respectively. For a comprehensive discussion of the Collatz function using *Mathematica* see also Chapter 7 of Vardi's book [5].

There are situations in which one needs to look at several possibilities (so `Which` would be a good tool) and if none of the possibilities occurred, then as a last resort, carry on with one specific case. This will be demonstrated in the next problem.

Problem 5.7

Define the function

$$f(x) = \begin{cases} -x, & \text{if } |x| < 1 \\ \sin(x), & \text{if } 1 \le |x| < 2 \\ \cos(x), & \text{otherwise.} \end{cases} \tag{5.1}$$

$\Longrightarrow$ Solution.

The function $f(x)$ is defined as follows: if $|x| < 1$ or $1 \le |x| < 2$ then $f(x) = -x$ or $f(x) = \sin(x)$, respectively. However, if x does not fall into any of the cases above, then $f(x)$ is defined as $\cos(x)$. Here is how we handle this using `Which`:

```
f[x_] := Which[
  Abs[x] < 1, -x,
  1 <= Abs[x] < 2, Sin[x],
  True, Cos[x]
  ]
```

As you guess, $|x|$ is translated into *Mathematica* using the `Abs` function. Here is a little test for the function `f`:

```
f /@ {0.5, Pi/2, Pi}
{-0.5, 1, -1}
```

There is also another way to define this function using the command `Piecewise`.

```
g[x_] := Piecewise[{{-x, Abs[x] < 1}, {Sin[x], 1 <= Abs[x] < 2}},
  Cos[x]]

g /@ {0.5, Pi/2, Pi}
{-0.5, 1, -1}
```

There is an important difference between these two definitions which will be explored in Problem 13.6. Basically, defining the function using `Which` makes *Mathematica* consider this function as a continuous function. We can rectify this problem using `Piecewise`.

Problem 5.8

Define the function

$$f(x,y) = \begin{cases} e^x & \text{if } x = y \\ \frac{e^x - e^y}{x-y} & \text{if } x \neq y \end{cases}$$

and observe that for arbitrary real numbers a, b such that $a < b < 0$ we have

$$\frac{f(x,b)}{f(x,a)} > \frac{1+e^b}{1+e^a}$$

for any $a \leq x \leq b$.

$\Longrightarrow$ Solution.

Here we also have a function with more than one definition: if $x = y$ then $f(x,y) = e^x$; otherwise $f(x,y) = \frac{e^x - e^y}{x-y}$. Thus the definition of this function calls for the `If` statement.

```
ec[x_, y_] := If[x == y, E^x, (E^x - E^y)/(x - y)]
```

We will use `Plot` to check the claim of the problem. For graphics, see Chapter 13.

```
a = 9; b = 10
Plot[{ec[x, b]/ec[x, a], (1 + E^b)/(1 + E^a)}, {x, a, b}]
```

Out[58]=

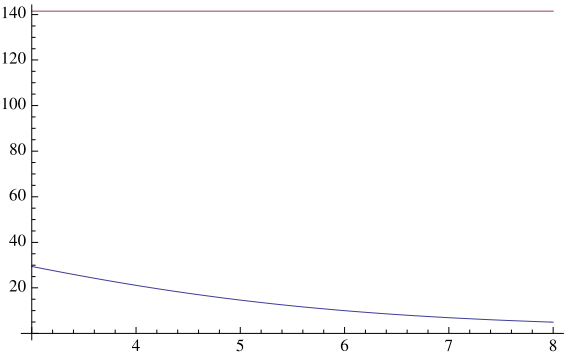


Figure 5.1 Graphs of $\frac{f(x,10)}{f(x,9)}$ and $\frac{1+e^{10}}{1+e^{9}}$

6
Sums and products

This chapter is devoted to evaluating series. *Mathematica* provides two commands, `Sum` and `Product` (and their numerical cousins, `NSum`, `NProduct`) to handle series.

In the previous chapters we could write a code to calculate the series

$$1+\frac{1}{1}+\frac{1}{2!}+\cdots+\frac{1}{n!}.$$

Wolfram *Mathematica*® offers us two commands, namely `Sum` and `Product`, to handle problems of this nature easily without getting into any programming complications.

6.1 Sum

Study the following examples:

```
Sum[s[i], {i, 1, 7}]
s[1] + s[2] + s[3] + s[4] + s[5] + s[6] + s[7]

Sum[s[i], {i, 1, k}]
```

$\sum_{i=1}^{k} s[i]$

The second example shows again that *Mathematica* can handle things symbolically.

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_6, © Springer-Verlag London Limited 2010

Problem 6.1

Write the function $ep(n) = 1 + \frac{1}{1} + \frac{1}{2!} + \cdots + \frac{1}{n!}$.

$\Longrightarrow$ Solution.

Here is the code:

```
ep[n_] := 1 + Sum[1/k!, {k, 1, n}]

N[ep[100]]
2.71828

N[E]
2.71828
```

Sometimes *Mathematica* can do great things:

```
ep[Infinity]
𝔼
```

This shows that the above series converges to $\mathbb{E}$.

Problem 6.2

Prove that

$$(1 + 2 + 3 + \cdots + n)^2 = (1^3 + 2^3 + 3^3 + \cdots + n^3).$$

$\Longrightarrow$ Solution.

Writing the above equality symbolically, we want to show $\sum_{i=1}^{n} i^3 = (\sum_{i=1}^{n} i)^2$. This example shows that *Mathematica* is aware of formulas for certain sums, including the ones above:

```
p[n_] := Sum[i, {i, 1, n}]

p[n]
(1/2) n (1 + n)

p3[n_] := Sum[i^3, {i, 1, n}]

p3[n]
(1/4) n^2 (1+n)^2

p[n]^2==p3[n]
True
```

The above example shows *Mathematica* knows that $1+2+\cdots+n=\frac{n(n+1)}{2}$ and $1^3+2^3+\cdots+n^3=(\frac{n(n+1)}{2})^2$. The first formula was known to Gauss at the age of seven. In fact he proved the formula as follows:

$$\begin{array}{cccc} 1 & 2 & \cdots & n \quad + \\ n & n-1 & \cdots & 1 \\ \hline n+1 & n+1 & \cdots & n+1 \end{array}$$

Thus twice the sum of the series is $n(n+1)$ and thus the formula. The second formula follows by an easy induction.

Exercise 6.1

Evaluate $\sum_{k=1}^{n} \frac{k}{k^4+k^2+1}$.

Problem 6.3

Write a function to calculate the following series

$$s(n)=\frac{1}{1}+\frac{1}{1+2}+...+\frac{1}{1+2+...+n}$$

$\Longrightarrow$ SOLUTION.

A glance at the series shows that there are in fact two series involved. Thus one needs two `Sum`, one to take care of $1+2+\cdots+i$ and the other for the sum of these expressions.

```
s[n_] := Sum[1/Sum[j, {j, 1, i}], {i, 1, n}]
```

One probably needs a few minutes to be convinced that this code generates the series given in the problem. One of the advantages of the Front-End in *Mathematica* is its ability to write mathematics as one writes on paper. Writing the above series using mathematical symbols, one has $\sum_{i=1}^{n}(1/\sum_{j=1}^{i} j)$ which is much more understandable than `s[n]`.

Using the palette provided by *Mathematica*, one can enter exactly the same expression in the front-end and define the function `s` this way.

$s[n_] = \sum_{i=1}^{n}(1/\sum_{j=1}^{i} j)$

Mathematica can easily handle complicated symbolic calculations as the following example demonstrates. Recall that the binomial coefficient $\binom{n}{k}$ stands for $\frac{n!}{k!(n-k)!}$. The command `Binomial[n, k]` is available (see Problem 1.5).

Problem 6.4

Define

$$p(n) = \sum_{k=0}^{n} \binom{n}{k}^2 (1+x)^{2n-2k}(1-x)^{2k}$$

and show that, for any chosen n, the coefficients of x are positive.

$\Longrightarrow$ Solution.

We shall first carefully translate the above formula into *Mathematica*.

```
p[n_] := Sum[Binomial[n, k]^2(1 + x)^(2n - 2k)(1 - x)^(2k),
{k, 0, n}]

p[3]
(1 - x)^6 + 9(1 - x)^4(1 + x)^2 + 9(1 - x)^2(1 + x)^4 + (1+x)^6

Expand[p[3]]
20 + 12x^2 + 12x^4 + 20x^6
```

As one sees, all the coefficients are positive. One can gather these coefficients in a list

```
CoefficientList[Expand[p[7]], x]

{3432, 0, 1848, 0, 1512, 0, 1400, 0, 1400, 0, 1512, 0, 1848, 0,
3432}
```

This is one of my favorite examples of using `Sum`.

Problem 6.5

Define $S(k,n) = \sum_{i=1}^{n} i^k$. Prove that

$$\sum_{a=0}^{n} \frac{S(2, 3a+1)}{S(1, 3a+1)} \tag{6.1}$$

is always a squared number.

$\Longrightarrow$ Solution.

The function `s` takes two variables and is defined as a series. Once we have defined this function, we can simply plug it into Equation 6.1.

```
s[k_, n_] := Sum[i^k, {i, 1, n}]

Sum[s[2, 3a + 1]/s[1, 3a +1],{a,0, n}]

1 + n + n (1 + n)

Factor[%]
(1 + n)^2
```

♣ TIPS

– `Sum` will try to evaluate the precise sum of the series. If an approximation suffices, use `NSum` which is often much faster.

We finish this section by observing "live" that the series

$$\frac{1}{1}+\frac{1}{4}+\frac{1}{9}+\frac{1}{25}+\cdots$$

conversges to $\frac{\pi^2}{6}$.

```
Manipulate[Pi^2 / 6 - NSum[1 / i^2, {i, 1, n}], {n, 1, 100 000, 1}]
```

Exercise 6.2

Define the functions $f(k)=\sum_{n=1}^{k}(-1)^n\frac{t^n}{n!}$ and $g(k)=\sum_{n=1}^{k}(-1)^n\frac{n!}{t^n}$. Show that

$$2-f(2)g(2)=\frac{2}{t}+\frac{t}{2}.$$

Exercise 6.3

Write the function

$$f(k)=\sin(x)+x+\frac{x^3}{1\times 2\times 3}+\frac{x^5}{1\times 2\times 3\times 4\times 5}+\cdots+\frac{x^k}{1\times 2\times\cdots\times k}$$

(k is odd).

Exercise 6.4

Investigate that

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} \sum_{k=0}^{2n} \frac{1}{2n+4k+3} = \frac{3\pi}{8} \log \frac{\sqrt{5}+1}{2} - \frac{\pi}{16} \log 5$$

Exercise 6.5

Investigate whether the series

$$1 + \frac{1}{2} + \frac{1}{3} - \frac{1}{4} - \frac{1}{5} - \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} - - - + + + \cdots$$

converges.

Exercise 6.6

Investigate whether the series

$$2^{-\frac{1}{2}} + (3+5)^{-\frac{1}{2}} + (7+11+13)^{-\frac{1}{2}} + (17+19+23+29)^{-\frac{1}{2}} + \cdots$$

converges.

6.2 Product

The command `Product` is very similar to `Sum`, but here instead of sum we have series involving products:

```
Product[s[i], {i, 1, 7}]
s[1]s[2]s[3]s[4]s[5]s[6]s[7]

Product[s[i], {i, 1, k}]
```

$\prod_{i=1}^{k} s[i]$

Here is a code to produce $(x + \frac{1}{x})(x^2 + \frac{1}{x^2}) \cdots (x^n + \frac{1}{x^n})$.

```
p[n_] := Product[(x^k + 1/x^k), {k, 1, n}]
```

Problem 6.6

Write the following series:

$$\frac{1 \times 3}{2 \times 2} \frac{3 \times 5}{4 \times 4} \frac{5 \times 7}{6 \times 6} \cdots$$

and show that this series tends to $2/\pi$.

$\Longrightarrow$ Solution.

First one needs to recognize that the general term of this series is

$$\frac{(2n-1)(2n+1)}{2n \times 2n},$$

i.e.,

$$\prod_{n=1} \frac{(2n-1)(2n+1)}{2n \times 2n} = \frac{1 \times 3}{2 \times 2} \frac{3 \times 5}{4 \times 4} \frac{5 \times 7}{6 \times 6} \cdots .$$

Knowing this, it is easy to write down the code

```
N[Product[(2 n - 1) (2 n + 1)/(2 n)^2, {n, 1, 100}] - 2/Pi]
0.0015856

N[Product[(2 n - 1) (2 n + 1)/(2 n)^2, {n, 1, 1000}] - 2/Pi]
0.000159095

N[Product[(2 n - 1) (2 n + 1)/(2 n)^2, {n, 1, 10000}] - 2/Pi]
0.0000159149
```

This shows that as n grows, the product gets closer to $\frac{2}{\pi}$. The punch line is:

```
Product[(2 n - 1) (2 n + 1)/(2 n)^2, {n, 1, \[Infinity]}]
2/Pi
```

♣ TIPS

– `Product` will try to evaluate the precise product of the series. If an approximation suffices, use `NProduct` which is often much faster.

Exercise 6.7

Let F_i be the i-th Fibonacci number. Write the function

$$f(n) = (F_1 + x)(F_2 + x^2) \cdots (F_n + x^n).$$

What is the coefficient of x^4 in $f(23)$? (Hint: see `Coefficient`.)

Exercise 6.8

Investigate that

$$\frac{\exp}{2} = (\frac{2}{1})^{\frac{1}{2}} (\frac{2}{3}\frac{4}{3})^{\frac{1}{4}} (\frac{4}{5}\frac{6}{5}\frac{6}{7}\frac{8}{7})^{\frac{1}{8}} (\frac{8}{9}\frac{10}{9}\frac{10}{11}\frac{12}{11}\frac{12}{13}\frac{14}{13}\frac{14}{15}\frac{16}{15})^{\frac{1}{16}} \cdots$$

7
Loops and repetitions

In this chapter we will look at traditional loops available in *Mathematica*, i.e., ways to repeat a block of code for a number of times. We will introduce `Do`, `While` and `For` loops and study nested loops, that is, loops defined inside each other. We finish the chapter by looking at other nested commands.

If we agree that the primary ability that a computer language provides is the ability to repeat a certain code "fast" then Wolfram *Mathematica*® provides three *loops* that enable us to repeat part of our codes. These are quite similar to the loops that exist in any procedural language like Pascal or C.

7.1 Do, For a While

The first and the simplest one is the `Do` loop. Here is the traditional example. The structure of the `Do` loop reminds one of the commands such as `Sum` or `Table`.

```
Do[Print[i],{i,1,7}]
1
2
3
4
5
6
7
```

The code:

```
Do[f[i],
{i,1,1000000}]
```

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_7, © Springer-Verlag London Limited 2010

repeats the expression `f[i]` one million times where `i` runs from 1 to 1000000. In fact this is (almost) equivalent to

```
f/@ Range[10000000]
```

Here is a little comparison.

```
Timing[Do[f[i],{i,1,10000000}]]
{6.93 Second,Null}

Timing[f/@ Range[10000000];]
{5.008 Second,Null}
```

Apart from writing a code which is faster, the art of programming is also to try to write codes in a way in which they are also readable.

We will write Problem 3.15 using a `Do` loop.

Problem 7.1

Find out how many primes bigger than n and smaller than $2n$ exist, when n runs from 1 to 15.

$\Longrightarrow$ Solution.

First we find all prime numbers up to 30.

```
prime30=Select[Range[30],PrimeQ]
{2,3,5,7,11,13,17,19,23,29}
```

Now for any n we check how many prime numbers lie between n and $2n$. To do this, as in Problem 3.15, we create a list of numbers between n and $2n$, `Range[n+1,2n-1]` then using `Intersection` we find out how many prime numbers are in this interval `Range[n+1,2n-1]` $\bigcap$ `prime60`. Using `Length` we can find the number of primes that lie in this interval. Once we have this, then using a `Do` loop, we run this code for n from 1 to 30.

```
Do[
Print[i, " ::::", Length[Intersection[Range[n + 1, 2 n - 1],
prime60]]],
{n, 1, 15}]
```

```
1::::0
2::::1
3::::1
4::::2

5::::1
6::::2
7::::2
8::::2
```

```
9::::3
10::::4
11::::3
12::::4
13::::3
14::::3
15::::4
```

For our next application of a `Do` loop, recall Problem 3.6. The formula $n^2 + n + 41$ produces prime numbers when n runs from 0 to 39. This was noticed by Euler some 300 years ago. One wonders whether one gets more consecutive prime numbers for a different constant in the above formula. The next problem examines this:

Problem 7.2

Consider the formula $n^2 + n + i$. Find out the number of consecutive primes (starting from $n = 0$) that one gets when i runs from 1 to 10000.

$\Longrightarrow$ Solution.

One way to approach the problem is to write a code to find out how many consecutive primes one gets (starting from $n = 0$) for a fixed i in the formula $x^2 + x + i$. Once this is done, then one can use a `Do` loop to change the value of i from 1 up to 10000. The code which finds out the number of consecutive primes is the same in nature as Problem 3.14.

The code

```
Select[Range[500], (PrimeQ[#^2 + # + 41] == False &), 1]
{40}
```

returns the first number in the range of $\{0, \cdots, 500\}$ such that the formula $n^2 + n + 41$ does not return a prime number. All we have to do now is to assemble this code in a loop as follows:

```
A={}
Do[
A=Union[A,Select[Range[500],(PrimeQ[#^2+#+i]==False&),1]],
{i,1,10000}];A

{1,2,3,4,5,6,7,10,12,16,40}
```

A line such as

```
A=Union[A,Select[Range[100],(PrimeQ[#^2+#+i]==False&),1]]
```

which is equivalent to

```
A= A ⋃ Select[Range[100],(PrimeQ[#^2+#+i]==False&),1]
```

collects all new results in the list `A`. We have seen this trick in Example 5.3.

A glance at the result shows that n^2+n+41 produces the maximum number of consecutive primes as was noticed by Euler. As a matter of fact, the formula which produces 16 consecutive prime numbers is $n^2 + n + 17$ which was also found by Euler!

Exercise 7.1

Modify the code of Problem 7.2 to find for which values of i one gets $\{10, 12, 16, 40\}$ consecutive prime numbers from the formula $n^2 + n + i$.

Problem 7.3

The sum of two positive integers is 5432 and their least common multiple is 223020. Find the numbers.

⟹ Solution.

```
Do[
 If[LCM[i, 5432 - i] == 223020, Print[i, "  ", 5432-i]],
 {i, 1, 2718}]

1652  3780
```

We shall see more examples of the `Do` loop later. The next loop is the `While` loop. This one operates on a Boolean (`True` or `False`) statement and gives you the ability to repeat a block until the Boolean statement becomes `False`.

Problem 7.4

Find the first prime number consisting only of ones and greater than 11.

⟹ Solution.

Here is the mystery code:

```
n=111;
While[!PrimeQ[n],
    n=10n+1];
Print[n]

111111111111111111
```

`!PrimeQ[n]` is our Boolean statement. Recall that `!` here stands for negation, or `Not` (see page 55). That is, if a statement, say `s`, is `True`, then `!s` becomes `False`. Here `n=10n+1` is the code we want to repeat. The code `n=10n+1` simply gets the number `n` and places 1 at the far right of the number (right?). So the aim is to put as many 1s in front of the original `n` which is 111 here to get a prime number. The `While` loop does exactly this. It is going to repeat the above code until `!PrimeQ[n]` becomes `False`. That is, until `PrimeQ[n]` becomes `True`, which happens when `n` becomes prime. And this is what we are looking for.

As you can see, the `While` loop has the form

```
While[condition,body].
```

The body of the loop can consist of several lines separated by `;`.

Here is a little test to see that the result of the above code is consistent with the code we wrote on page 38.

```
IntegerDigits[n]
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}

Length[%]
19
```

Problem 7.5

Find the smallest positive integer m such that $529^3 + 132^3 m$ is divisible by 262417.

$\Longrightarrow$ Solution.

We start with $m = 1$ and while the remainder on division of $529^3 + 132^3 m$ by 262417 is not zero, in *Mathematica* terms, `While[Mod[529^ 3 + 132^ 3m, 262417] != 0`, then we add one to m, i.e., `m++`, and repeat this until the remainder is zero. Then this is the m we are looking for:

```
m = 1;
While[Mod[529^3 + 132^3 m , 262417] != 0, m++]; m
1984
```

Note that `m++` is equivalent to `m=m+1`, which adds one to `m`.

One can also use `Select` in the spirit of Problem 3.14 to get the result (if one knows a bound for m) as follows:

```
Select[Range[10000], Mod[529^3 + 132^3 # , 262417] == 0 &, 1]
{1984}
```

Problem 7.6

Find the closest prime number less than a given number n.

$\Longrightarrow$ Solution.

Here we have an example which can be "naturally" written by `While`. Notice that the body of the loop contains one line.

```
n = Input["enter a number"]
While[! PrimeQ[n],
    n--];
Print[n]
```

`Input` opens a box and asks for a value. This is a good way if one wants to ask a user for data. Again `!PrimeQ[n]` returns `True` and keeps the loop repeating until `n` is prime. That's what the question asks.

Problem 7.7

Find all prime numbers less than a given n.

$\Longrightarrow$ Solution.

We will use the loop `While` to find one by one all the prime numbers smaller than n starting from the smallest prime number 2. Notice that here the body of `While` has two sentences.

```
i = 1; n = Input["enter a number"]; pset = {};
While[Prime[i] ≤ n,
  pset = pset ∪ {Prime[i]};
  i++];
pset
```

Ok, for $n = 321$ we get all the prime numbers up to 321.

```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59,
61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131,
137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197,
199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271,
277, 281, 283, 293, 307, 311, 313, 317}
```

Here, until `Prime[i]` is smaller than `n`, the loop keeps collecting `Prime[i]` in a list `pset` which at the beginning we set empty (see Example 5.3). After each step we go a step forward by adding one to i, that is `i++`, and repeat the same procedure again until `Prime[i]` is bigger than `n`.

Note that one could use `AppendTo` instead of `pset = pset ∪ {Prime[i]}` as follows: `AppendTo[pset,Prime[i]]` (see page 59 for more discussion on `AppendTo`).

Exercise 7.2

Find the smallest positive multiple of 99999 that contains no 9's amongst its digits. (Hint, see `MemberQ`.)

Problem 7.8

Determine the highest power of 5 that divides $\dfrac{(5n)!}{(n!)^5}$ for $1 \leq n \leq 200$.

$\Longrightarrow$ Solution.

```
A = {}; Do[i = 0;
 While[Mod[(5 n)!/(n!)^5, 5^i] == 0,
  AppendTo[A, {n, i}]; i = i + 1],
 {n, 1, 200}];

 Max[Last /@ A]
12

Select[A, #[[2]] == 12 &]
{{124, 12}}
```

The result shows that, for $n = 124$, 5^{12} divides $\frac{(5n)!}{(n!)^5}$ and this is the highest power. The code consists of two loops, one `Do` loop to change the value of n and a `While` loop inside it to determine what powers of 5 divide $\frac{(5n)!}{(n!)^5}$. This is an example of a nested loop that we will study in Section 7.2.

The last loop in *Mathematica* is the `For` loop. Here is the easiest example:

```
For[i=5,i<10,i++,Print[i]]

5
6
7
8
9
```

The loop `For` consists of different parts as follows:

`For[init,condition,steps,body]`.

The `init` part is where we initialize the variables we need to use in the body of the loop. In the above example this was `i=5`. The second part is where a Boolean expression appears and is where we decide when to terminate the loop. The last part is reserved for the body of the loop. Each of these parts can have several sentences which should be separated by `;`. Let us look at another example.

Problem 7.9

Find the sum of the sequence

$$\frac{1}{1+2}+\frac{2}{2+3}+\cdots+\frac{10}{10+11}.$$

$\Longrightarrow$ Solution.

```
For[i = 1; sum = 0, i < 11, i++, sum += i/(i + i + 1)];
sum
64157087/14549535
```

Notice that the `init` part of the loop consists of two lines. Also notice that `sum+=i/(i + i + 1)` is a shorthand for `sum=sum+i/(i + i + 1)` as `i++` is a shorthand for `i=i+1`. In the same way `i*=n` is a shorthand for `i=i*n`.

To refresh the memory, here are the other approaches to get the sum of the above sequence

```
Sum[i/(2i + 1), {i, 1, 10}]
64157087/14549535

Plus @@ (#/(2# + 1) & /@ Range[10])
64157087/14549535
```

One can leave out any part of a `For` loop. For example

```
For[ ,False , , Print["Never see the light of day"]]
```

produces nothing. One can also see that `While[test,body]` is the same as `For[ ,test, , body]` and `Do[body,{x,xmin,xmax,inc}]` is the same as `For[x=xmin,x≤xmax,x+=inc,body]`. But again, there are times when `While` makes the code more readable and there are times when `For` is a better choice.

Let us do some experiments:

```
Timing[Do[,{10^6}]]
{0.02 Second,Null}

Timing[Do[,{1000000}]]
{0.1 Second,Null}

Timing[i=1;While[i<10^6,i++]]
{2.614 Second,Null}

Timing[i=1;While[i<1000000,i++]]
{1.932 Second,Null}

Timing[For[i=1,i<10^6,i++]]
{2.654 Second,Null}

Timing[For[i=1,i<1000000,i++]]
{1.973 Second,Null}
```

Here is one more example.

Problem 7.10

An integer $d_n d_{n-1} d_{n-2} \dots d_1$ is *palindromic* if

$$d_n d_{n-1} d_{n-2} \dots d_1 = d_1 d_2 \dots d_{n-1} d_n$$

(for example 15651). Write a code to ask for a number $d_n d_{n-1} d_{n-2} \dots d_1$ and find out if it is palindromic. Enhance the code further such that if the number is not palindromic then the code tests whether $d_n d_{n-1} d_{n-2} \dots d_1 + d_1 d_2 \dots d_{n-1} d_n$ is (for example, 108+801=909). Furthermore write a code to give the number of times it is needed to repeat this procedure until one gets a palindromic number starting with $d_n d_{n-1} d_{n-2} \dots d_1$ (if it takes more than 150 times, let the function return infinity).

$\Longrightarrow$ Solution.

We start with an example. Let $n = 98$. We need to check systematically whether n is palindromic. If not, then produce 89, add this to $n = 98$ and check whether this is palindromic. We have seen how to produce the reverse of a number using `IntegerDigits, Reverse` and `FromDigits` (see Problem 3.10). Here is the first step:

```
n=98;nlist=IntegerDigits[n]
{9,8}

If[nlist != Reverse[nlist],n=n+FromDigits[Reverse[nlist]]]
187
```

If the result is not palindromic, one has to do the same procedure again. Thus we use a `While loop` to do this for us.

```
n = 98; nlist = IntegerDigits[n];
While[nlist != Reverse[nlist],
  n = n + FromDigits[Reverse[nlist]];
  nlist = IntegerDigits[n]
  ]; n

8813200023188
```

One can enhance the code:

```
n = Input["Enter a number"]; i = 1; nlist = IntegerDigits[n];
safetyNet = True;

While[nlist != Reverse[nlist] && safetyNet,
  Print[i, " ", n]; n = n + FromDigits[Reverse[nlist]]; i++;
  nlist = IntegerDigits[n];
  If[i > 150, safetyNet = False]
  ]
If[i > 150, Print["........Aborted"], Print[i, " ", n]
  ]
```

7.2 Nested loops

In many applications there are several factors (variables) which change simultaneously, and this calls for what we call a *nested loop.* Instead of trying to describe the situation, let us see some examples.

```
Do[
  Do[
    Print[i, " ", j],
    {j, 1, 2}
    ],
  {i, 1, 3}
  ]
```

```
1 1
1 2
2 1
2 2
3 1
3 2
```

The code contains two `Do` loops, one inside the other. In the inner one, the counter `j` runs from 1 to 2 and once this is done, the outer loop performs and the counter `i` goes one further and again the inner loop starts to run.

Problem 7.11

Find all the pairs (n, m) for $n, m \leq 10$ such that $n^2 + m^2$ is a squared number (e.g., $(3, 4)$ as $3^2 + 4^2 = 5^2$).

$\Longrightarrow$ Solution.

```
Do[
  Do[
   If[Sqrt[i^2 + j^2] ∈ Integers, Print[i, " ",
j]],
   {j, i, 10}
  ],
{i, 1, 10}
]
```

Here is the result

```
3  4
6  8
```

Here the outer loop starts with the counter `i` getting the value 1. Then it is the turn of the block inside this loop, which is again another loop run. In the inner loop `{j, i, 10}` makes the counter `j` run from `i` to 10. This done, in the outer loop `i` takes 2 and then `j` runs from 2 to 10 and so on and each time checks whether $\sqrt{i^2 + j^2}$ is an integer. The reader should see that this is enough to find all the pairs up to 10 with the desired property. Can you say how many times the `If` line is going to be performed?

One can make the nested `Do` loop a bit shorter. The following is an equivalent code to the first example of a Nested `Do` loop on page 81.

```
Do[
 Print[i , " ", j],
 {i, 1, 3}, {j, 1, 2}]

1 1
1 2
2 1
2 2
3 1
3 2
```

Note that here `j` is the counter for the inner loop.

We have already seen the command `Table` which provides a sort of loop. In fact, `Table` can provide us with a nested loop as well.

```
Table[{i, j}, {i, 1, 3}, {j, 1, 2}]
{{{1, 1}, {1, 2}}, {{2, 1}, {2, 2}}, {{3, 1}, {3, 2}}}
```

One should compare this with the example of the nested `Do` loop above. As the result shows, here `j` is the counter for the inner loop. One of the issues that might arise here is that the output is a nested list (i.e., too many { and }). Sometimes we really do not need the nested list answer to our question. For example, we want to come up with a code to solve Problem 7.11 by using `Table`. In order to get rid of extra "{", one can use the command `Flatten`.

```
Flatten[Table[{i, j}, {i, 1, 3}, {j, 1, 4}]]
{1, 1, 1, 2, 1, 3, 1, 4, 2, 1, 2, 2, 2, 3, 2, 4, 3, 1, 3, 2, 3,
3, 3, 4}
```

`Flatten` gets rid of all the lists inside a list, i.e., removes all the "{". In our problem we want a list of pairs. In this case we need

```
Flatten[Table[{i, j}, {i, 1, 3}, {j, 1, 4}],1]
{{1, 1}, {1, 2}, {1, 3}, {1, 4}, {2, 1}, {2, 2}, {2, 3}, {2, 4},
{3, 1}, {3, 2}, {3, 3}, {3, 4}}
```

Now we have all the pairs. But some of them are repeated. For us `{1,3}` is the same as `{3,1}`. So, as in Problem 7.11, we need the inner counter to depend on the outer one as follows:

```
Flatten[Table[{i, j}, {i, 1, 3}, {j, i, 4}], 1]
{{1, 1}, {1, 2}, {1, 3}, {1, 4}, {2, 2}, {2, 3}, {2, 4}, {3, 3},
{3, 4}}
```

All we have to do now is to select the pairs (m, n) such that $\sqrt{m^2 + n^2} \in \mathbb{N}$.

```
Select[Flatten[Table[{i, j}, {i, 1, 10}, {j, i,
10}], 1],
(Sqrt[#[[1]]^2 + #[[2]]^2] ∈ Integers) &]


{{3, 4}, {6, 8}}
```

Exercise 7.3

A pair (m, n) such that $m^2 + n^2$ is a squared number is called a Pythagorean pair (see Problem 7.11). Find a Pythagorean pair (m, n) such that if the digits of m are written in reverse order, then n is obtained.

Exercise 7.4

Pick an odd prime number p. Then find a pair (q, r) of positive integers such that $p^2 + q^2 = r^2$.

7.3 Nest, NestList and more

Let $f(x)$ be a function defined on a variable x. There are times when one needs to apply the function f to itself several times, i.e., $f(\cdots f(f(x))\cdots)$ (see the examples in page 20). *Mathematica* provides a command to do exactly this:

```
Nest[f, x, 4]
f[f[f[f[x]]]]
```

If one wants to keep track of each step, the command `NestList` is available

```
NestList[f, x, 4]
{x, f[x], f[f[x]], f[f[f[x]]], f[f[f[f[x]]]]}
```

Here are some (nice) examples:

```
f[x_]:=1/(1+x)
Nest[f,x,4]
```

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}}$$

```
NestList[f,x,4]
```

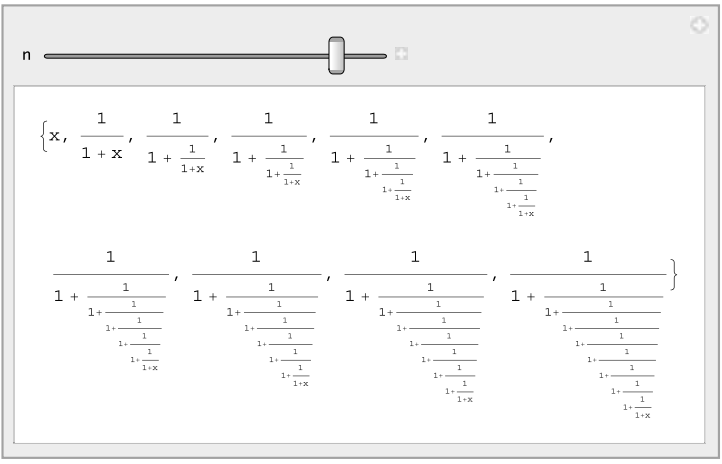
$$\left\{\frac{1}{1+x}, \frac{1}{1+\frac{1}{1+x}}, \frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}, \frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}}\right\}$$

```
NestList[Sqrt[#+6]&,Sqrt[6],4]
```

$$\{\sqrt{6}, \sqrt{6+\sqrt{6}}, \sqrt{6+\sqrt{6+\sqrt{6}}}, \sqrt{6+\sqrt{6+\sqrt{6+\sqrt{6}}}}, \sqrt{6+\sqrt{6+\sqrt{6+\sqrt{6+\sqrt{6}}}}}\}$$

Using the dynamic variable n we can monitor how the continuous function "grows" as n runs from 1 to 10.

```
f[x_] := 1 / (1 + x)

Manipulate[NestList[f, x, n], {n, 1, 10, 1}]
```



Exercise 7.5

Find the roots of the equation

$$x = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cdots \cfrac{1}{1+x}}}}$$

where there are 10 division lines in the expression on the right. (Hint, use `Solve` for finding roots, more on this in Chapter 14.)

Exercise 7.6

Investigate that

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2}\frac{\sqrt{2+\sqrt{2}}}{2}\frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}\cdots$$

There are two more commands of this type, `NestWhile` and `NestWhileList`.

```
?NestWhile
NestWhile[f, expr, test] starts with expr, then
repeatedly applies f until  applying test to the
result no longer yields True.
```

The following problem uses `NestWhile`.

Problem 7.12

A happy number is a number such that if one squares its digits and adds them together, and then takes the result and squares its digits and adds them together again and keeps doing this process, one comes down to the number 1. Find all the happy ages, i.e., happy numbers up to 100.

$\Longrightarrow$ Solution.

```
Select[Range[100],
  NestWhile[
        Plus @@ (IntegerDigits[#]^2)&,#,(!#==4) && (!#==1)&]==1&]

{1,7,10,13,19,23,28,31,32,44,49,68,70,79,82,86,91,94,97,100}
```

Deciphering this code is a bit challenging. Note that the code contains three pure functions (so, three `&` for those) and a boolean expression containing `And` (so `&&` for `And`).

There is a more elegant approach to this problem using recursive functions in Problem 11.4. In any case, one can observe that happy ages are mostly before one gets a job or after retirement!

Now, we are going to make up a problem and use `NestList` to get some answers.

Problem 7.13

A number $a_1a_2\cdots a_n$ is called a pure prime number if

$$a_1a_2\ldots a_n,\ a_1a_2\ldots a_{n-1},\ldots,a_1a_2a_3,\ a_1a_2,\ \text{and}\ a_1$$

are all prime. Prove that pure prime numbers are finite in number and find all of them.

$\Longrightarrow$ Solution.

First we have to find a way to drop the last digit of a number. The function `Quotient` might help

```
?Quotient

Quotient[m, n] gives the integer quotient of m and n.

Quotient[5937, 10]
593

Quotient[593, 10]
59

Quotient[59, 10]
5
```

The above example shows that applying `Quotient` to a number repeatedly drops the last digit of the number one by one. Thus

```
NestList[Quotient[#, 10] &, 5937, 3]

{5937,593,59,5}
```

Now we have all the numbers. We only need to test whether all of them are prime.

```
PrimeQ /@ NestList[Quotient[#,10]&,5937,3]
{False, True, True, True}
```

Thus 5937 just misses being a pure prime. If we want to define this as a function, a little problem might arise. In the case of 5937 we have to apply `Quotient` three times to this number. But for a number n with arbitrary digits, we need to use `FixedPointList`.

```
?FixedPointList

FixedPointList[f, expr] generates a list giving the results of
applying f  repeatedly, starting with expr, until the results no
longer change.

FixedPointList[Quotient[#,10]&,5937]
{5937, 593, 59, 5, 0, 0}

FixedPointList[Quotient[#,10]&,7647653]
{7647653, 764765, 76476, 7647, 764, 76, 7, 0, 0}
```

It is clear that we have to drop the last two elements from the list.

```
Drop[FixedPointList[Quotient[#,10]&,5937],-2]
{5937, 593, 59, 5}
```

Now it is time to apply `PrimeQ` to the list to check whether all these numbers are prime.

```
PrimeQ[Drop[FixedPointList[Quotient[#,10]&,5937],-2]]
{False,True,True,True}
```

What we need is a list containing only `True`'s. Thus if only one of the numbers happens to be not prime, the whole number is not pure prime as is the case with 5937. We can combine all the Booleans in the list with `And` and the result would make it clear whether the number is pure prime. Here is the code

```
Apply[And,{False,True,True,True}]
False
```

Thus, putting all these together, we have

```
purePrime[n_]:=Apply[And,PrimeQ[Drop[FixedPointList[
Quotient[#,10]&,n],-2]]]

Select[Range[10, 99], purePrime]
{23,29,31,37,53,59,71,73,79}

Select[Range[100,999],purePrime]
{233, 239, 293, 311, 313, 317, 373, 379, 593, 599, 719, 733, 739,
797}

Select[Range[1000, 9999], purePrime]
{2333,2339,2393,2399,2939,3119,3137,3733,3739,3793,3797,5939,
7193,7331,7333,7393}
```

This seems not to be a good algorithm to find all pure prime numbers as it already takes some time to find all the 6-digit pure primes. In fact the problem here is that in order to find, say, all the 4-digit pure primes, the above algorithm has to check all the numbers from 1000 to 9999. But this is not necessary. The following example demonstrates this. If we know that 719 is pure prime then all we have to check, to find the pure primes which have four digits and whose first three digits are 719, are the numbers $\{7190, 7191, ..., 7199\}$.

```
Range[719*10, 719*10 + 9]
{7190,7191,7192,7193,7194,7195,7196,7197,7198,7199}
```

We do not need to consider even numbers.

```
Range[719*10+1, 719*10 + 9,2]
{7191,7193,7195,7197,7199}
```

Now we need to find out which of these numbers are prime.

```
Select[%, PrimeQ]
{7193}
```

This shows that 7193 is a pure prime. Thus we start with all one-digit primes and find all the two-digit primes as above.

```
purelist={2,3,5,7}
{2, 3, 5, 7}

Range[10#+1,10#+9,2]&[purelist]
{{21, 23, 25, 27, 29}, {31, 33, 35, 37, 39}, {51, 53, 55, 57, 59},
{71, 73, 75, 77, 79}}

Flatten[Range[10#+1,10#+9,2]&[purelist]]
{21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 51, 53, 55, 57, 59, 71,
73, 75, 77, 79}

purelist=Select[Flatten[Range[10#+1,10#+9,2]&[purelist]],PrimeQ]
{23, 29, 31, 37, 53, 59, 71, 73, 79}
```

Thus these are all the two-digit pure primes. Having them, we can immediately find all three-digit primes.

```
purelist=Select[Flatten[Range[10#+1,10#+9,2]&[purelist]],PrimeQ]
{233,239,293,311,313,317,373,379,593,599,719,733,739,797}
```

Thus our clever code to find all the pure primes is as follows:

```
purelist={2,3,5,7};
While[purelist != {},
  purelist=Select[Flatten[Range[10#+1,10#+9,2]&[purelist]],PrimeQ];
  Print[purelist]
  ]

{23,29,31,37,53,59,71,73,79}

{233,239,293,311,313,317,373,379,593,599,719,733,739,797}

{2333,2339,2393,2399,2939,3119,3137,3733,3739,3793,3797,5939,
7193,7331,7333,7393}

{23333,23339,23399,23993,29399,31193,31379,37337,37339,37397,
59393,59399,71933,73331,73939}

{233993,239933,293999,373379,373393,593933,593993,719333,739391,
739393,739397,739399}

{2339933,2399333,2939999,3733799,5939333,7393913,7393931,7393933}

{23399339,29399999,37337999,59393339,73939133}
```

♣ TIPS

– In Problem 7.13 we used the command `FixedPointList`. There are also two other commands in the same spirit, namely, `LengthWhile` and `TakeWhile`.

Exercise 7.7

Starting with a number, consider the sum of all the proper divisors of the number. Now consider the sum of all the proper divisors of this new number and repeat this process. If one eventually obtains the number which one started with, then this number is called a *social number*. Write a program to show that 1264460 is a social number.

7.4 Fold and FoldList

Recall one of the questions we asked in Chapter 3, namely: Given $\{x_1, x_2, \cdots, x_n\}$ how can one produce $\{x_1, x_1 + x_2, \cdots, x_1 + x_2 + \cdots + x_n\}$?

Let us look at the commands `Fold` and `FoldList`.

```
Fold[f,x,{a,b,c}]
f[f[f[x,a],b],c]

FoldList[f,x,{a,b,c}]
{x,f[x,a],f[f[x,a],b],f[f[f[x,a],b],c]}
```

Replace the function `f` with `Plus` and `x` with `0` and observe what happens. (See the following problem for the answer.) Here is a use of `FoldList` to write another code for Problem 6.3.

Problem 7.14

Write a function to calculate the sum of the following sequence.

$$p(n) = \frac{1}{1} + \frac{1}{1+2} + ... + \frac{1}{1+2+...+n}$$

$\Longrightarrow$ Solution.

Here is the code:

```
p[n_]:= Plus @@ (1/Rest[FoldList[Plus, 0, Range[n]]])
```

In order to decipher this code, let us look at the standard example of `FoldList`.

```
FoldList[Plus,0,{a,b,c}]
{0,a,a+b,a+b+c}
```

Thus, dropping the annoying 0 from the list:

```
 Rest[FoldList[Plus,0,{a,b,c}]]
{a,a+b,a+b+c}
```

and

```
1/Rest[FoldList[Plus,0,{a,b,c}]]
```

$$\left\{\frac{1}{a}, \frac{1}{a+b}, \frac{1}{a+b+c}\right\}$$

makes the original code clear.

We have used `FoldList` in its simplest form, that is,

```
FoldList[Plus,0,{a,b,c}]
{0,a,a+b,a+b+c}
```

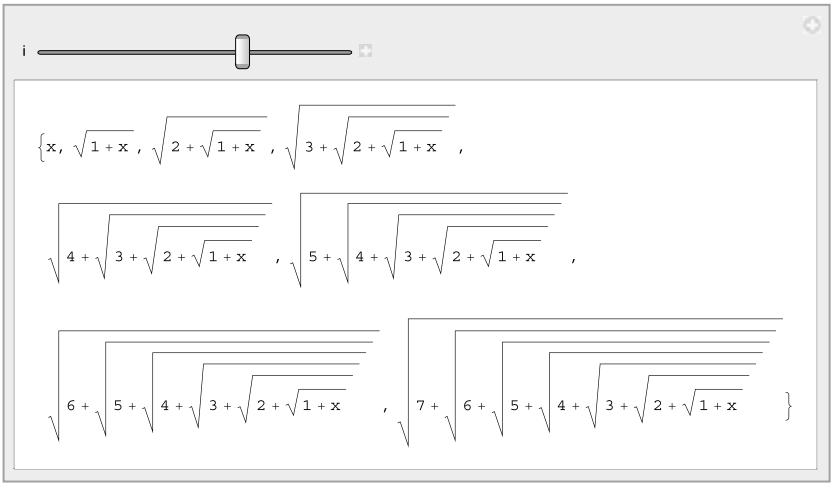
If we are just after this, namely, "a list of the successive accumulated totals of elements in a list", then the command `Accumulate` does just that as well:

```
Accumulate[{a, b, c, d}]
{a, a + b, a + b + c, a + b + c + d}
```

Later in Problems 13.8 and 13.10 we will use these commands to generate very interesting graphs.

Here is one more example where the function used in `FoldList` is not `Plus`.

```
Manipulate[FoldList[Sqrt[#1 + #2] &, x, Range[i]], {i, 1, 10, 1}]
```



Problem 7.15

For which natural numbers n is it possible to choose signs $+$ and $-$ in the expression

$$1^2 \pm 2^2 \pm 3^2 \pm \cdots \pm n^2$$

so that the result is 0?

$\Longrightarrow$ Solution.

One can find the following code in Vardi [5].

```
Fold[(#1/.x→x+#2)(#1/.x→x-#2)&,x,{a,b,c}]/.x→1

(1-a-b-c) (1+a-b-c) (1-a+b-c) (1+a+b-c) (1-a-b+c)
(1+a-b+c) (1-a+b+c) (1+a+b+c)
```

Motivated by this code, one can approach the problem.

```
Do[ If[(Fold[(#1/.x→x+#2)(#1/.x→x-#2)&,x,
Range[n]^2]/.x→0) = 0,Print[n] ],
{n,1,40}]
7 8 11 12 15 16 19 20 23
$ Aborted
```

However, this seems to take time and there might be a better way to approach this problem.

Here is another approach:

```
 t[n_]:=Flatten[Outer[List,Sequence @@Table[{I
k,k}^2,{k,2,n}]],n-2]

Do[
  If[Select[t[n],Plus @@ #==-1&,1]!={},Print[n]],
  {n,3,40}]

7
8
11
12
15
16
19
20

Hold[Abort[],Abort[]]
```

Is there a better way to do this?

7.5 Inner and Outer

Recall the following questions from Chapter 3: Given $\{x_1, x_2, x_3, \cdots, x_n\}$ and $\{y_1, y_2, y_3, \cdots, y_n\}$, how can one produce the combinations

$$\{x_1, y_1, x_2, y_2, x_3, y_3, \cdots, x_n, y_n\}$$

and

$$\big\{\{x_1, y_1\}, \{x_1, y_2\}, \{x_1, y_3\}, \cdots, \{x_1, y_n\} \\ \{x_2, y_1\}, \{x_2, y_2\}, \cdots, \{x_n, y_1\}, \{x_n, y_2\}, \cdots, \{x_n, y_n\}\big\}?$$

One can get the first list using the command `Inner` and the second list by using `Outer`:

```
Inner[f,{a,b},{x,y},g]
g[f[a,x],f[b,y]]
```

If we replace the functions `f` and `g` with `List` we get:

```
Inner[List,{a,b},{x,y},List]
{{a,x},{b,y}}

Flatten[%]
{a,x,b,y}
```

Or this one to get rid of `Flatten`:

```
Inner[Sequence,{a,b},{x,y},List]
{a,x,b,y}
```

There is a more clever way to get this using `Transpose` (see Chapter 12).

```
Flatten[Transpose[{{a, b}, {x, y}}]]
{a, x, b, y}
```

Now for getting the second list:

```
Outer[f, {a, b}, {x, y, z}]
{{f[a, x], f[a, y], f[a, z]}, {f[b, x], f[b, y], f[b, z]}}

Outer[List, {a, b}, {x, y, z}]
{{{a, x}, {a, y}, {a, z}}, {{b, x}, {b, y}, {b, z}}}

Flatten[Outer[List, {a, b}, {x, y, z}], 1]
{{a, x}, {a, y}, {a, z}, {b, x}, {b, y}, {b, z}}
```

As one can imagine, commands of this type provide many possibilities and inspiring compositions of functions!

Problem 7.16

Consider the pairs (m, n) where $1 \leq m, n \leq 30$. We want to consider a graph with vertices the numbers 1 to 30 and an *edge* from m to n if mn is a prime number. For example, there is an edge between 3 and 1 as 31 is a prime number. Draw this graph.

$\Longrightarrow$ Solution.

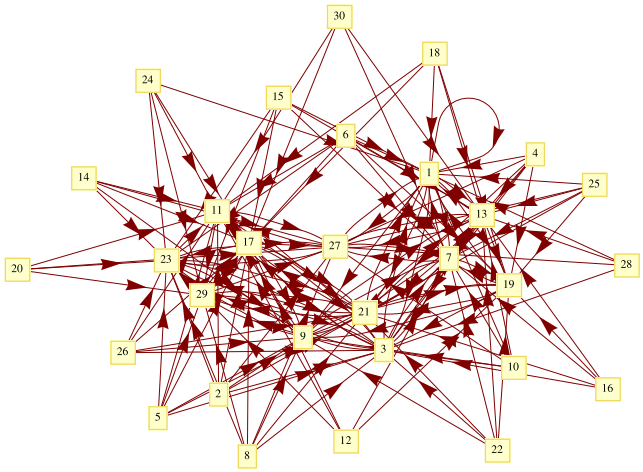
We first produce all the pairs (m, n) where $1 \leq m, n \leq 30$. This can be easily done with `Outer` as follows:

```
s = Flatten[Outer[List, Range[30], Range[30]], 1]
```

Then we consider each pair (m, n), check if mn is prime, if this is the case we then collect $m \to n$ (using `Rule`). This is because of how the command `GraphPlot` works.

```
t = If[PrimeQ[FromDigits[#]], Rule @@ #, 0] & /@ s;
k = DeleteCases[t, 0];
GraphPlot[k, VertexLabeling -> True, DirectedEdges -> True]
```

Out[260]=



Exercise 7.8

Write a code to convert $\{x_1, x_2, \ldots, x_k, x_{k+1}\}$ to

$$\{x_1 \to x_2, x_2 \to x_3, \ldots, x_k \to x_{k+1}\}.$$

Exercise 7.9

Consider a number. Then sort the decimal digits of this number in ascending and descending order. Subtract these two numbers (for example, starting from 5742, we get $7542 - 2457 = 5085$). This is called the Kaprekar routine. First check that starting with any 4-digit number and repeating the Kaprekar routine, you always reach either 0 or 6174. Then find out among all the 4–digit numbers, what is the maximum number of iterations needed in order to get 6174.

8
Substitution, *Mathematica* rules

This chapter introduces a way to substitute an expression by another expression without changing their values. This is done by setting a substitution rule for *Mathematica* to follow. This simple idea provides a way to write very elegant programs.

In Wolfram *Mathematica*® one can substitute an expression with another using *rules*. In particular, one can substitute a variable with a value without assigning the value to the variable. Here is how it goes:

```
x + y /. x -> 2
2 + y
```

This means we replace x with 2 without assigning 2 to x. If we ask for the value of x, we see

```
x
x

FullForm[x + y /. x -> 2]
Plus[2,y]
```

The following examples show the variety of things one can do with rules.

```
x + y /. {x -> a, y -> b}
a + b

x^2 - 3 /. x -> Sqrt[3]
0

x^2 + y /. x -> y /. y -> x
x + x^2
```

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_8, © Springer-Verlag London Limited 2010

```
Solve[x^3 - 2 x + 1 == 0]
{{x -> 1}, {x -> 1/2 (-1 - Sqrt[5])},
{x ->  1/2 (-1 + Sqrt[5])}}

x /. %
{1, 1/2 (-1 - Sqrt[5]), 1/2 (-1 + Sqrt[5])}

x + 2 y /. {x -> y, y -> a}
2 a + y

FullForm[x + 2 y /. {x -> y, y -> a}]
Plus[Times[2,a],y]
```

The last example reveals that *Mathematica* goes through the expression only once and replaces the rules. If we need *Mathematica* to go through the expression again and repeat the whole process until no further substitution is possible, one uses `//.` as follows:

```
x + 2 y //. {x -> y, y -> a}
3 a
```

In fact `/.` and `//.` are shorthand for `ReplaceAll` and `ReplaceRepeated` respectively.
Be careful not to make *Mathematica* and yourself confused:

```
ReplaceRepeated[x + 2 y, {x -> y, y -> x}]

During evaluation of In[11]:= ReplaceRepeated::rrlim: Exiting
after x+2 y scanned 65536 times. >>
x + 2 y
```

One can use `MaxIterations` to instruct *Mathematica* how many times to repeat the substitution process. This comes in quite handy in many of the problems we look at in this book:

```
ReplaceRepeated[1/(1 + x), x -> 1/(1 + x), MaxIterations -> 4]

During evaluation of In[14]:= ReplaceRepeated::rrlim: Exiting
after 1/(1+x) scanned 4 times. >>
```

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}}$$

If you don't want to get the annoying message of ".... scanned 4 times", use `Quiet` to prevent messages of these kind.

```
Quiet[ReplaceRepeated[1/(1 + x), x -> 1/(1 + x),
MaxIterations -> 4]]
```

$$\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}}$$

In Chapter 9 we will use the rules effectively with the pattern matching facility of *Mathematica* (see, for example, Problem 9.3).

Problem 8.1

Generate the following list $\{x^{y^z}, x^{z^y}, y^{x^z}, y^{z^x}, z^{x^y}, z^{y^x}\}$.

$\Longrightarrow$ Solution.

The command `Permutations` generates a list of all possible permutations of the elements. Once we get all the possible arrangements, we can use a rule to change $\{x, y, z\}$ to x^{y^z} as shown below:

```
Permutations[{x, y, z}]
{{x, y, z}, {x, z, y}, {y, x, z}, {y, z, x}, {z, x, y}, {z,
  y, x}}

Permutations[{x, y, z}] /. {a_, b_, c_} -> a^b^c
{x^y^z, x^z^y, y^x^z, y^z^x, z^x^y, z^y^x}
```

Exercise 8.1

Explain what the following code does:

```
(a + b)^n /. (x_ + y_)^z_ -> (x^z + y^z)
```

Problem 8.2

To get the Thue–Morse sequence, start with 0 and then repeatedly replace 0 with 01 and 1 with 10. So the first four numbers in the sequence are

0

01

0110

01101001

Write a function to produce the n-th element of this sequence. (We will visit this sequence again in Problem 13.10.)

$\Longrightarrow$ Solution.

This problem is just waiting for the *Mathematica* rules to get into action! We define a general rule to replace 0 with 01 and replace 1 with 10. However, to do so, we consider 0 and 1 in a list, and define the rule as `0->{0,1}` and

`1->{1,0}` and, at the end, we get rid of all the parentheses with `Flatten` and put all digits together again using `FromDigits`, as the following codes show:

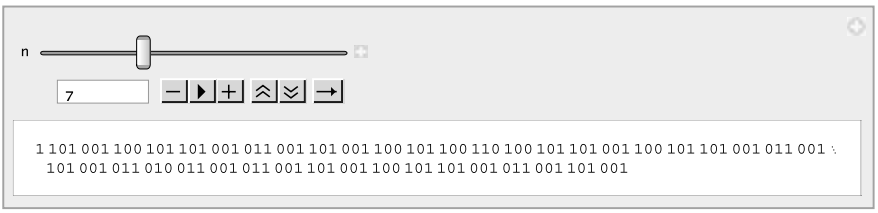
```
{0} /. {0 -> {0, 1}, 1 -> {1, 0}}
{{0, 1}}

{{0, 1}} /. {0 -> {0, 1}, 1 -> {1, 0}}
{{{0, 1}, {1, 0}}}

{{{0, 1}, {1, 0}}} /. {0 -> {0, 1}, 1 -> {1, 0}}
{{{{0, 1}, {1, 0}}, {{1, 0}, {0, 1}}}}

Flatten[%]
{0, 1, 1, 0, 1, 0, 0, 1}

FromDigits[%]
1101001
```

So we define a function, repeating the above process using `ReplaceRepeated` and control the number of the iterations by `MaxIterations` as in the example on page 97.

```
morse[n_] :=
 Quiet[FromDigits[
  Flatten[ReplaceRepeated[{0}, {0 -> {0, 1}, 1 -> {1, 0}},
    MaxIterations -> n]]]]

morse[4]
110100110010110

morse[6]
1101001100101101001011001101001100101100110100101101001
100110010110
```

Note that *Mathematica* drops the first 0 that Thue–Morse starts with (as it considers this as a number). To avoid this, one can replace 0 with x and 1 with y and then the replacement rules are `x->{x,y}` and `y->{y,x}`.

To see how quickly this series grows, wrap the code with `Manipulate` and run n from 1 to 20 as follows: (Note that we have used `Quiet` as in page 97 to prevent any messages being produced because of the use of `MaxIterations`.)



9
Pattern matching

Everything in *Mathematica* is an expression and each expression has a pattern. One can search for a specific pattern and change it to another pattern. This is called pattern matching programming. This chapter explains this method. Get ready to be amazed!

Everything in Wolfram *Mathematica*® is an expression and each expression has a pattern. *Mathematica* provides us with the ability to decide whether an expression matches a specific pattern. Following R. Gaylord's exposition [1], consider the expression x^2. This expression is precisely of the following form or *pattern*, "x raised to the power of two":

```
MatchQ[x^2, x^2]
True
```

But x^2 will be matched also by the following loose description, "something" or "an expression"

```
MatchQ[x^2, _]
True
```

Here `_` stands (or rather sits) for an expression (`_` is called *a blank* here). Also x^2 will match "x to the power of something"

```
MatchQ[x^2, x^_]
True
```

Before we go further, we need to mention that one can give a name to a blank expression as follows `n_`. Here the expression `_` is labelled `n`. (We have already seen `n_` in defining a function. In fact, when defining a function, we label an expression that we plug into the function.) Also one can restrict the expression by limiting its head! Namely, `_head` matches an expression with the head `head`. Look:

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_9, © Springer-Verlag London Limited 2010

```
FullForm[x^2]
Power[x, 2]

Head[x^2]
Power

MatchQ[x^2, _Power]
True

Head[4]
Integer

MatchQ[4,_Integer]
True

Head[4/3]
Rational

MatchQ[4/3,_Integer]
False
```

Putting these together, `n_Plus` means an expression which is labelled `n` and has the head `Plus`. Continuing with our example, x^2 matches "x to the power of an integer number"

```
MatchQ[x^2, x^_Integer]
True

MatchQ[x^2, x^_Real]
False
```

The same way x^2 matches "something or an expression to the power of 2"

```
MatchQ[x^2, _^2]
True

MatchQ[x^2, _^5]
False
```

Finally, x^2 matches "something to the power of something"

```
MatchQ[x^2,_^_]
True
```

One can put a condition on a pattern, namely to test whether an expression satisfies a certain condition. Here is an example:

```
MatchQ[5, _Integer?(# > 3 &)]
True

MatchQ[2, _Integer?(# > 3 &)]
False
```

The pattern `_Integer?(# > 3 &)` stands for an expression which has `Integer` as its head, i.e., an integer number, and is greater than three.

Here are more examples:

```
MatchQ[x^2, _^_?OddQ]
False

MatchQ[x^2, _^_?EvenQ]
True
```

Here is how all these concepts help. One can single out an expression of specific pattern and once this is done then change the expression. It is all about accessing and then manipulating!

Here are some examples:

```
MatchQ[{a,b},{_,_}]
True

MatchQ[{a,b},{x_,y_}]
True
```

Study the following examples carefully!

```
{{a, b}, {c, d}} /. {x_, y_} -> x y
{a c, b d}

{{a, b}, {c, d}} /. {x_, y_} -> x^ y
{a^c, b^d}

{{a, b}, {c, d}} /. {x_, y_} -> y
{c, d}
```

Here is the third approach to Problems 3.14 and 4.3.

Problem 9.1

Write a function `squareFreeQ[n]` that returns `True` if the number n is a square free number, and `False` otherwise.

$\Longrightarrow$ Solution.

```
t=FactorInteger[234090]
{{2,1},{3,4},{5,1},{17,2}}
```

We are after those numbers that when decomposed into powers of primes, say, $\{\{p_1, k_1\}, \{p_2, k_2\}, \cdots, \{p_t, k_t\}\}$, then all k_i are 1.

The pattern `{_,_?(#>1&)}` describes those lists with the second element (a number) bigger than 1.

```
MatchQ[{3,4},{_,_?(#>1&)}]
True
```

Here is the time to introduce `Cases`.

```
? Cases

Cases[{e1, e2, ... }, pattern] gives a list of the ei that match
the pattern.

Cases[{6,test,20,5.3,35,5/3},_Integer]
{6,20,35}
```

We use `Cases` to get all the pairs with k_i greater than 1. If this list is not empty then the number is not square free.

```
Cases[{{2,1},{3,4},{5,1},{17,2}},{_,_?(#>1&)}]
{{3,4},{17,2}}

Cases[{{2,1},{3,4},{5,1},{17,2}},{_,_?(#>1&)}] == {}
False
```

We are ready to put all these together and write a function for finding square free numbers.

```
squareFree3[n_]:=Cases[FactorInteger[n],{_,_?(#>1&)}] == {}

squareFree3[234090]
False

squareFree3[3*5*13*17]
True
```

As with `Select` (see Problem 3.14), with `Cases` it is also possible to get only the first n expressions that satisfy a given pattern.

```
Cases[expr,pattern->rhs,levelspec] gives the values of rhs that
match the pattern.

Cases[expr,pattern,levelspec,n] gives the first n parts in expr
that match the pattern. >>
```

As in our problem, we only need to find one case of k_i being greater than 1, so it is enough to use the `Cases` and single out just one of these items (again compare this with Problem 3.14).

```
squareFree3[n_]:=Cases[FactorInteger[n],{_,_?(#>1&)},1,1] == {}

squareFree3[234090]
False

squareFree3[3*5*13*17]
True
```

So far we have been dealing with one expression. What if, instead of one expression, we would be dealing with a bunch of them?

```
MatchQ[{x^2},{_}]
True

MatchQ[{x^2, x^3, x^5}, {_}]
False

MatchQ[{x^2, x^3, x^5}, {__}]
True
```

As one can see from the above example, __ stands for a sequence of data as _ is for just one expression. In fact __ is for a sequence of nonempty expressions, and ___ is for a sequence of empty or more data. The following examples show this clearly.

```
MatchQ[{},{_}]
False

MatchQ[{},{__}]
False

MatchQ[{},{___}]
True
```

Here is one more example to show the difference between __ and ___:

```
MatchQ[{3,5,2,2,stuff,7},{__,3,___}]
False

MatchQ[{3,5,2,2,stuff,7},{___,3,___}]
True

MatchQ[{3,5,2,2,7,us},{___,2,2,___}]
True

{one, two, three, four} /. {x_, y___, z_} -> {z, y^z, y/z}
{four, two^three^four, (three two)/four}
```

Exercise 9.1

Explain what the following code does.

```
datasample = Table[Random[Integer, {1, 10}], {1000}];
frequence[data_List, n_] :=
  Apply[Plus, data /. n -> "a" /. x_Integer -> 0 /. "a" -> 1];
Table[frequence[datasample, n], {n, 1, 10}]
```

Problem 9.2

In the first 50000 prime numbers, find those that have 2009 embedded in them (e.g., 420097 is prime and 2009 is sitting in it).

$\Longrightarrow$ Solution.

Here are two quite similar solutions: the second solution uses a technique similar to Exercise 9.1.

```
(* First solution *)

Select[IntegerDigits /@ Prime /@ Range[1, 50000],
 MatchQ[#, {m___, 2, 0, 0, 9, n___}] &]

{{3, 2, 0, 0, 9}, {5, 2, 0, 0, 9}, {8, 2, 0, 0, 9}, {9, 2, 0, 0,
  9}, {1, 2, 0, 0, 9, 1}, {1, 2, 0, 0, 9, 7}, {1, 7, 2, 0, 0,9},
  {1, 8, 2, 0, 0, 9}, {2, 0, 0, 9, 0, 3}, {2, 0, 0, 9, 0, 9},
  {2, 0, 0, 9, 2, 7}, {2, 0, 0, 9, 2, 9}, {2, 0, 0, 9, 7, 1},
  {2, 0, 0, 9, 8, 3}, {2, 0, 0, 9, 8, 7}, {2, 0, 0, 9, 8, 9},
  {2, 4, 2, 0, 0, 9}, {2, 7, 2, 0, 0, 9}, {3, 0, 2, 0, 0, 9},
  {3, 2, 2, 0, 0, 9}, {3, 3, 2, 0, 0, 9}, {4, 2, 0, 0, 9, 7},
  {4, 4, 2, 0, 0, 9}, {4, 5, 2, 0, 0, 9}, {5, 1, 2, 0, 0, 9},
  {5, 3, 2, 0, 0, 9}}

(* second solution *)
 t =
 Select[IntegerDigits /@
    Prime /@ Range[1, 50000] /. {m___, 2, 0, 0, 9, n___} ->
    {X, m, 2, 0, 0, 9, n}, MemberQ[#, X] &]

{{X, 3, 2, 0, 0, 9}, {X, 5, 2, 0, 0, 9}, {X, 8, 2, 0, 0,
  9}, {X, 9, 2, 0, 0, 9}, {X, 1, 2, 0, 0, 9, 1},
  {X, 1, 2, 0, 0, 9, 7}, {X, 1, 7, 2, 0, 0, 9},
  {X, 1, 8, 2, 0, 0, 9}, {X, 2, 0, 0, 9, 0, 3},
  {X, 2, 0, 0, 9, 0, 9}, {X, 2, 0, 0, 9, 2, 7},
  {X, 2, 0, 0, 9, 2, 9}, {X, 2, 0, 0, 9, 7, 1},
  {X, 2, 0, 0, 9, 8, 3}, {X, 2, 0, 0, 9, 8, 7},
  {X, 2, 0, 0, 9, 8, 9}, {X, 2, 4, 2, 0, 0, 9},
  {X, 2, 7, 2, 0, 0, 9}, {X, 3, 0, 2, 0, 0, 9},
  {X, 3, 2, 2, 0, 0, 9}, {X, 3, 3, 2, 0, 0, 9},
   {X, 4, 2, 0, 0, 9, 7}, {X, 4, 4, 2, 0, 0, 9},
   {X, 4, 5, 2, 0, 0, 9}, {X, 5, 1, 2, 0, 0, 9},
   {X, 5, 3, 2, 0, 0, 9}}

FromDigits /@ Rest /@ t
{32009, 52009, 82009, 92009, 120091, 120097, 172009,
182009, 200903, 200909, 200927, 200929, 200971, 200983, 200987,
200989, 242009, 272009, 302009, 322009, 332009, 420097, 442009,
452009, 512009, 532009}
```

Exercise 9.2

Observe that `Range[10]/.{x_,y___}→ y/x` amounts to `10!`

We are ready to write a little game.

Problem 9.3

Write a game as follows. A player gets randomly 7 cards between 1 and 9. He would be able to drop any two cards between 4 and 9 that are the same. Then the sum of the cards that remain in the hand is what a player scores. A player with minimum score wins.

$\Longrightarrow$ Solution.

First, we generate a list containing 7 random numbers between 1 and 10.

```
s=Table[RandomInteger[{1, 9}], {7}]
{2,5,2,3,4,7,4}
```

Now we shall write a code to discard any two numbers which are the same. The trick we use here is, we look inside the list and recognize the same numbers (which have the same pattern), mark them and with a rule send the list to a new list containing all the elements except the similar ones. Here is the code:

```
s /. {m___, x_, y___, x_, n___} -> {m, y, n}
{5,3,4,7,4}
```

Here *Mathematica* looks for similar expressions `x_` and `x_`. To the left of `x_` is `m___` which means any sequence of empty or more expressions. Similarly in between `x`'s we place `y___`. That is, in between similar numbers could be empty (i.e., the similar numbers could be next to each other) or a bunch of other expressions. Finally to the right-hand side of the second `x_` is `n___`. In the example, our original list is `{2,5,2,3,4,7,4}`. *Mathematica* recognizes that there are two 2's in the list, so will assign them to `x_`. To the left-hand side of the first 2 there is no data, thus `m___` would get an empty value, `y___` would be 5 and `n___` would be the whole sequence of $3, 4, 7, 4$ to the right-hand side of the second 2. Thus the rule`{m___,x_,y___,x_,n___}->{m,y,n}` will discard the `x`'s and give us `{5,3,4,7,4}`.

Still there are two 4's in the list but, as we have seen in Chapter 8, `/.->` would go through the list only once. Thus if we run the same code again, this time with our new list, we shall get rid of double 4's.

```
{5, 3, 4, 7, 4} /. {m___, x_, y___, x_, n___} -> {m, y, n}
{5, 3, 7}
```

Remember that `//.->` was designed exactly for this job.

```
s //. {m___, x_, y___, x_, n___} -> {m, y, n}
{5,3,7}
```

But in the game we are allowed to drop the cards between 4 and 10. Thus we shall put in a little test to find numbers larger than 3 which are the same. Here is the enhanced code:

```
s //. {m___, x_?(3 < # < 10 &), y___, x_, n___} -> {m, y, n}
{2,5,2,3,7}
```

Notice that it is enough to put a test for one of the `x_`'s.
Just to make sure we understood this, let's try the code for

```
s = {7, 6, 2, 7, 1, 2, 1, 6, 7};

s //. {m___, x_?(3 < # < 10 &), y___, x_, n___} -> {m, y, n}
{2, 1, 2, 1, 7}
```

It remains to sum the numbers in the list. For example

```
Plus @@ %
13
```

Problem 9.4

A *Kaprekar* number is a number that if it is squared then the representation of the square can be split into two (positive) integer parts whose sum is equal to the original number (e.g. 45, since $45^2 = 2025$, and $20 + 25 = 45$). Find all 5-digit Kaprekar numbers.

$\Longrightarrow$ Solution.

Let us start with the number 45. We shall look at $45^2 = 2025$, then consider $2 + 025$, $20 + 25$ and $202 + 5$. Thus we first write a program to create a list of all these arrangements. For this we need the command `ReplaceList` which is similar in spirit to the commands `Replace` and `ReplaceRepeated` which we saw in Chapter 8.

```
?ReplaceList

ReplaceList[expr,rules] attempts to transform the entire
expression expr by applying a rule or list of rules in all
possible ways, and returns a list of the results obtained


ReplaceList[{2, 0, 2, 5}, {x__, y__} -> {{x}, {y}}]
{{{2}, {0, 2, 5}}, {{2, 0}, {2, 5}}, {{2, 0, 2}, {5}}}
```

We are almost there (except there are more curly brackets there than we would like). If we `Map` the command `FromDigits`, which puts these digits together and returns a number, we get all the arrangements we are looking for. We need to push `FromDigits` inside the second list, that is, it needs to be applied in the second level of the list we have, for this we use `Map[f,list,2]` where that 2 tells *Mathematica* to apply the map `f` to the second level in the list.

```
Map[FromDigits,
 ReplaceList[{2, 0, 2, 5}, {x__, y__} -> {{x}, {y}}], {2}]
{{2, 25}, {20, 25}, {202, 5}}
```

The rest is (almost) clear. If we replace the inside lists with `Plus` (namely, changing heads) we get the sum of all the numbers and this is exactly what we are after. Again, we want to apply `Plus` to the second layer (or level) of the list, for this, as in `Map`, we need to use `Apply[f,expr,2]` where that 2 refers to the second level. Remember the shorthand for `Apply` (to the first layer) is `@@` and for applying to the second layer is `@@@`, so we have

```
Plus @@@ (Map[FromDigits, ReplaceList[{2, 0, 2, 5},
{x__, y__} -> {{x}, {y}}], {2}])
{27, 45, 207}
```

This already shows 45 is a Kaprekar number as we see the number 45 in the list above. We are ready to write the whole list:

```
Select[Range[10000,99999],
 MemberQ[Plus @@@ (Map[FromDigits,
      ReplaceList[
       IntegerDigits[#^2], {x__, y__} -> {{x}, {y}}], {2}]), #] &]

{10000, 17344, 22222, 38962, 77778, 82656, 95121, 99999}
```

Recall that the Kaprekar numbers are those that one can write as *positive* integer parts whose sum is equal to the original number. This excludes 10, 100 and 10000 from this list as $10^2 = 100$ and $10 = 10+0$ but 0 is not positive. The reader is encouraged to modify the code to remove the cases of this nature.

Exercise 9.3

Find all the words in the *Mathematica* dictionary which end with "rat".

Problem 9.5

Find all the words in the *Mathematica* dictionary which contain the letters "c","u", "t" and "e".

$\Longrightarrow$ Solution.

```
cute = Select[DictionaryLookup[],
   MatchQ[Characters[#], {___, "c", ___, "u", ___, "t", ___,
      "e", ___}] &];
```

```
Short[cute]

{accentuate,accentuated, accentuates, <<700>>, woodcutter,
woodcutters}

Length[cute]
705
```

One needs to be careful when using pattern matching as the following problem demonstrates.

Problem 9.6

Write the Syracuse function which is defined as follows: for any odd positive integer n, $3n+1$ is even, so one can write $3n+1 = 2^k n'$ where k is the highest power of 2 that divides $3n+1$. Define $f(n) = n'$. Show that for any odd positive integer m, applying f repeatedly, we arrive at 1.

$\Longrightarrow$ Solution.

It is not difficult to see that this is in fact a different version of the Collatz function (see Problems 5.6 and 10.2). Here is one way to write the function. Recall that `FactorInteger` gives the decomposition of a number into its prime factors, i.e., if $n = 2^{k_1}3^{k_2}\cdots {p_i}^{k_i}$, then using `FactorInteger` we get $\{\{2,k_1\},\{3,k_2\},\cdots,\{p_i,k_i\}\}$. So, if we drop $\{2,k_1\}$ from the list and multiply the rest together, the result is the n' that we are looking for. Here is one way to do so:

```
FactorInteger[2^5*3*5*7]
{{2, 5}, {3, 1}, {5, 1}, {7, 1}}

Rest[FactorInteger[2^5*3*5*7]]
 {{3, 1}, {5, 1}, {7, 1}}

Rest[FactorInteger[2^5*3*5*7]] /. {x_, y_} -> x^y
{3, 5, 7}

Times @@ (Rest[FactorInteger[2^5*3*5*7]] /. {x_, y_} -> x^y)
105

f[n_?OddQ] := Times @@ (Rest[FactorInteger[3n+1]] /.
{x_, y_} -> x^y)

f[2^5*3*5*7]
105
```

Now that `f` is defined, we are going to apply `f` repeatedly to a given number until we reach one. For this we will use `FixedPointList` introduced in

Problem 7.13.

```
FixedPointList[f, 133]
{133, 25, 19, 29, 11, 17, 13, 5, 1, 1}
```

So far, so good. Let us start with 123.

```
FixedPointList[f, 123]

During evaluation of In[196]:= General::ovfl: Overflow occurred
in computation. >>

During evaluation of In[196]:= FactorInteger::exact: Argument
Overflow[] in FactorInteger[Overflow[]] is not an exact number.

During evaluation of In[196]:= FactorInteger::argt:FactorInteger
called with 0 arguments; 1 or 2 arguments are expected. >>

Out[196]= {123, 72759576141834259033203125,
68212102632969617843627 93, 51159076974727213382720 95,
76738615462090820074081 43, 11510792319313623011112215,
17266188478970434516668323, 25899282718455651775002485,
Overflow[], 1,  1}
```

What seems to be the problem? Let us look at the rule we have defined:

```
{{5, 1}, {37, 1}, {3, 2}} /. {x_, y_} -> x^y
{5, 37, 9}
```

This is what we wanted. Let us try another example:

```
{{5, 1}, {37, 1}} /. {x_, y_} -> x^y
{72759576141834259033203125, 1}
```

Something goes horribly wrong here. What we wanted is {5, 37}. The problem is, in the last example, since the list itself contains two lists, then x_ takes {5,1} and y_ takes {37,1}. To help *Mathematica* to get out of this confusion, we can describe the pattern of x_ more precisely, namely, x_ stands for an integer and not a list.

```
{{5, 1}, {37, 1}} /. {x_Integer, y_} -> x^y
{5, 37}
```

This is the correct approach. We redefine the function f according to this new rule.

```
Clear[f]

f[n_?OddQ] := Times @@ (Rest[FactorInteger[3n+1]] /.
{x_Integer, y_} -> x^y)

FixedPointList[f, 123]
{123, 185, 139, 209, 157, 59, 89, 67, 101, 19, 29, 11, 17,
13, 5, 1, 1}

FixedPointList[f, 133]
{133, 25, 19, 29, 11, 17, 13, 5, 1, 1}
```

Finally let us mention that there is another way to define the function using `IntegerExponent`:

```
?IntegerExponent
IntegerExponent[n,b] gives the highest power of b that
divides n. >>

f[n_] := Quotient[3 n + 1, 2^IntegerExponent[3 n + 1, 2]]

f[2^5*3*5*7]
105

FixedPointList[f, 123]
{123, 185, 139, 209, 157, 59, 89, 67, 101, 19, 29, 11, 17,
13, 5, 1, 1}
```

10
Functions with multiple definitions

In this chapter we will talk about functions with multiple definitions. Also we will see how a function can contain more than one line, that is, have a block of codes with its own local variables.

In Chapter 2, we defined functions in Wolfram *Mathematica*® which consisted of one line. In Section 5.3, we could define functions which came with conditions, using `If`, `Which` and `Piecewise`.

In this chapter we will talk about the ability of *Mathematica* to handle a function with multiple definitions. Also we will see how a function can contain more than one line, namely contain a block of codes (a sort of mini-program or a procedure).

Recall the very first function that we defined in Section 2.1.

```
f[n_] := n^2 + 4

f[13]
173
```

In the light of the previous chapter, one can see exactly what this code means. One can send any expression with any pattern into `f[n_]`. The expression is labelled `n`. Now we can easily restrict the sort of data we want to send into a function, by simply describing the sort of pattern we desire. For example if in the above function, we would like the function only to take on positive integers, then

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_10, © Springer-Verlag London Limited 2010

```
f[n_Integer?Positive] := n^2 + 4

f[4]
20

f[-2]
f[-2]
```

Here are some more examples:

```
g[n_Integer?(0 < # < 5 &)] := Sqrt[5 - n]

g[1]
2

g[6]
g[6]

g[2.6]
g[2.6]

e[p__?(PolynomialQ[#, x] &)] := Expand[p, x]

e /@ {4, (1 + x)^2, (1 + y)^2, (Sin[x] + Cos[x]^2)}
{4, 1 + 2 x + x^2, (1 + y)^2, e[Cos[x]^2 + Sin[x]]}
```

One can even be carried away with this ability. Here is a function that gives the prime factors of a number which consists only of odd digits (e.g. 3715).

```
myfunc[n_Integer?(Select[IntegerDigits[#], EvenQ, 1] == {}&)]:=
Map[First, FactorInteger[n]]

myfunc[3715]
{5, 743}

myfunc[593183]
myfunc[593183]
```

One of the great features of *Mathematica* is that one can define a function with multiple definitions. Here is a harmless example

```
oddeven[(n_?EvenQ)?Positive] := Print[n, " even and positive"]
oddeven[(n_?EvenQ)?Negative] := Print[n, " even and negative"]
oddeven[(n_?OddQ)?Positive] := Print[n, " odd and positive"]
oddeven[(n_?OddQ)?Negative] := Print[n, " odd and negative"]

Map[oddeven, {-2, 5, -3, -4}];
-2  even and negative
5 odd and positive
-3  odd and negative
4  even and positive
```

Here we have the function `oddeven` with four definitions. An integer falls into one of the cases above, and *Mathematica* has no problem going through all

the definitions of the function and applying the appropriate one to the given number. If one asks for the definition of `oddeven`, one can see *Mathematica* has all four definitions in memory, in the same order that one has defined the function.

```
?oddeven

Global‘oddeven

oddeven[(n_?EvenQ)?Positive] := Print[n,  even and positive]

oddeven[(n_?EvenQ)?Negative] := Print[n,  even and negative]

oddeven[(n_?OddQ)?Positive] := Print[n,  odd and positive]

oddeven[(n_?OddQ)?Negative] := Print[n,  odd and negative]
```

Problem 10.1

Define the function

$$f(x) = \begin{cases} \sqrt{x} & \text{if } x \geq 0 \\ \sqrt{-x} & \text{if } x < 0 \end{cases}$$

and plot the graph of the function for $-1 \leq x \leq 1$.

$\Longrightarrow$ Solution.

One can define $f(x)$ in *Mathematica* as a function with two definitions as follows:

```
f[x_?Positive] := Sqrt[x]
f[x_?Negative] := Sqrt[-x]
Plot[f[x],{x,-1,1}]
```

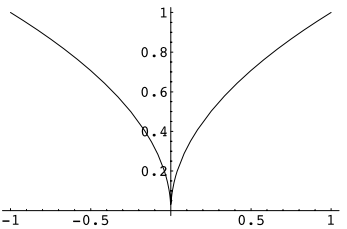


Figure 10.1 A function with multiple definitions

See Chapter 13 for more on graphics.

As you might have noticed so far, there has been no confusion regarding the multiple definitions of a function. Namely, the data that is sent to the function satisfied only one of the patterns in the definition of the function. In `oddeven`, a number could only be one of the cases of positive/negative and odd/even and in the previous example a number is either positive or negative. But imagine we define a function as follows:

```
f[x_] := x
f[x_Integer] := x!
```

Now one might ask what would be `f[4]`. There are two definitions for `f` and 4 can match both patterns, namely `x_ or x_Integer`.

```
f[4]
24

f[5]
120

f[2.3]
2.3

f[test]
test
```

Thus for any integer the definition which is the factorial of a number is performed and for other data the other definition (obviously). If we find out in what order *Mathematica* saves the definitions of functions, we can justify this action.

```
?f

Global‘f

f[x_Integer] := x
f[x_] := x
```

Thus in principle, *Mathematica* stores the definitions from the one with more precise pattern matching (here the one with `x_Integer`). If she cannot decide which definition has the more precise pattern matching, then she stores the definition in the order in which it has been entered into the system.

Problem 10.2

Define the Collatz function as follows:

$$f(x) = \begin{cases} x/2 & \text{if } x \text{ is even} \\ 3x+1 & \text{if } x \text{ is odd.} \end{cases}$$

Find out how many times one needs to apply f to numbers 1 to 200 to get 1.

$\Longrightarrow$ Solution.

We have defined this function in Problem 5.6 using `If`. Another version of this problem was discussed in Problem 9.6.

Here we simply use a multi-definition function to define the Collatz function.

```
f[x_Integer?EvenQ] := x/2

f[x_Integer] := 3x + 1
```

As we mentioned, the natural question which would be raised here is what is `f[2]`, as both definitions of `f` can accept this number. If cases like this happen, *Mathematica* chooses the definition which has the more precise description for the given expression. In the above definitions, although 2 is an integer and so fits in `_Integer`, however `_Integer?EvenQ` is a more accurate description for 2 and thus *Mathematica* chooses `f[x_Integer?EvenQ] := x/2` for evaluating `f[2]`. If *Mathematica* cannot decide in this manner, she would simply use the function which was defined first.

One can write the following one-liner for the rest of the code.

```
l=Length /@ ( NestWhileList[f, #, ! # == 1 &] & /@ Range[200])

{1, 2, 8, 3, 6, 9, 17, 4, 20, 7, 15, 10, 10, 18, 18, 5, 13, 21,
21, 8, 8, 16,  16, 11, 24, 11, 112, 19, 19, 19, 107, 6, 27, 14,
14, 22, 22, 22, 35,9, 110,  9, 30, 17, 17, 17, 105, 12, 25, 25,
25, 12, 12, 113, 113, 20, 33, 20, 33, 20,  20, 108, 108, 7, 28,
28, 28, 15, 15, 15, 103, 23, 116, 23, 15, 23, 23, 36,  36, 10,
23, 111, 111, 10, 10, 31, 31, 18, 31, 18, 93, 18, 18, 106, 106,
13, 119, 26, 26, 26, 26, 26, 88,13, 39, 13, 101, 114, 114, 114,
70, 21, 13, 34,  34, 21, 21, 34, 34, 21, 96, 21, 47, 109, 109,
109, 47, 8, 122, 29, 29, 29,  29, 29, 42, 16, 91, 16, 42, 16,
16, 104, 104, 24, 117, 117, 117, 24, 24, 16,  16, 24, 37, 24,
86, 37, 37,37, 55, 11, 99, 24,24, 112, 112, 112, 68,11, 50, 11,
125, 32, 32, 32, 81, 19,32, 32, 32, 19, 19, 94, 94, 19, 45, 19,
45,  107, 107, 107, 45, 14, 120, 120, 120, 27, 27, 27, 120, 27}

Max[%]

125

ListPlot[l, Filling -> Axis]
```

In this example, *Mathematica* stores the definitions of the function in the same order that we entered it, as there is no preference in the patterns that have been defined.

```
?f

Global`f

f[x_Integer?EvenQ]:=x/2

f[x_Integer]:=3 x+1
```

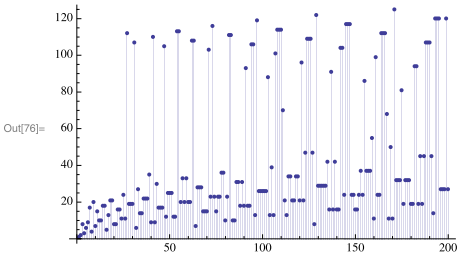


Figure 10.2 The Collatz function

Problem 10.3

Define a function $f(x)$ in *Mathematica* which satisfies

$$f(xy) = f(x) + f(y)$$
$$f(x^n) = nf(x)$$
$$f(n) = 0$$

where n is an integer and show that $f\big(\prod_{i=1}^{20} i(x_i)^i\big) = \sum_{i=1}^{20} if(x_i)$.

$\Longrightarrow$ Solution.

We first translate the function into *Mathematica*.

```
f[x_ y_] := f[x] + f[y]
f[x_^n_Integer] := n f[x]
f[n_Integer] = 0
```

```
f[Product[i(x_i)^i, {i, 1, 20}]]
f[x_1] + 2 f[x_2] + 3 f[x_3] + 4 f[x_4] + 5 f[x_5]
+ 6 f[x_6] + 7 f[x_7] + 8 f[x_8] + 9 f[x_9] + 10
f[x_10] + 11 f[x_11] + 12 f[x_12] + 13 f[x_13] + 14
f[x_14] + 15 f[x_15] + 16 f[x_16] + 17 f[x_17] + 18
f[x_18] + 19 f[x_19] + 20 f[x_20]
```

```
∑_i^20 i f[x_i]== f[Product[i (x_i)^i, {i, 1, 20}]]
True
```

10.1 Functions with local variables

One of the approaches of procedural languages (like C) to programming is to break the program into "mini-programs" or *procedures* and then put them together to get the code we need. These procedures have their own variables called *local* variables, that is, variables which have been defined only inside the procedure. So far all the functions that we have defined consist of only one line. *Mathematica*'s functions can also be used as procedures, namely can contain several lines of code and their own local variables. Let us look at a simple example. Recall Problem 7.7, which finds all the prime numbers less than n. Let us write this as a function `lPrimes[n]` to produce a list of all such primes.

```
lPrimes[n_] := Module[{pset = {}, i = 1},
While[Prime[i] <= n,
pset = pset ∪ {Prime[i]};
i++];
pset]

lPrimes[8]
{2, 3, 5, 7}
```

A function with several lines of code in *Mathematica* is wrapped by `Module`. The structure looks like `Module[{local variables},body]`. In the above example the variables `pset` and `i` are variables defined only inside the function `lPrimes`. Here to check that these are undefined outside the function:

```
pset
pset

i
i
```

♣ TIPS

– There are two other ways in *Mathematica* to collect codes, define local variables and make a mini-progam, namely `With` and `Block`. Compare these with `Module` using the *Mathematica* Document Center.

10.2 Functions with conditions

Consider the following code.

```
f[n_] := Sqrt[n] /; n > 0

f[4]
2

f[-4]
f[-4]
```

Here `/;` is a shorthand for `If`. We have seen that we can restrict the pattern of the data we pass into a function. The *almost* equivalent ways to define the above function are

```
f1[n_?Positive]:=Sqrt[n]

f2[n_]:=If[n>0,Sqrt[n]]
```

The following shows what the difference between these functions is:

```
f[4]
2

f1[4]
2

f2[4]
2

f[-3]
f[-3]

f1[-3]
f1[-3]

f2[-3]
```

Namely, the one which is defined by `If` would return `Null` if the argument does not satisfy the condition.

Sometimes using `/;` helps to make the code much more readable than using other ways to put conditions.

Here is another version of the game in Problem 9.3.

Problem 10.4

Write a game as follows. A player gets randomly 7 cards between 1 and 9. He would be able to drop any two cards with the sum 5. Then the sum of the cards that remain in the hand is what a player scores. A player with minimum score wins.

$\Longrightarrow$ Solution.

Let us first design a function that accepts a sequence of numbers and deletes any two numbers of which the sum is 5. Having an eye on the code of Problem 9.3 we proceed as follows:

```
aHandD[n___, y_, t___, z_, m___] :=
  aHandD[n, t, m] /; y + z == 5

aHandD[2,3,5,4,1,3,7]
aHandD[5,3,7]
```

The function is defined as follows: it looks inside the list of arguments (here `2,3,5,4,1,3,7`) and identifies the numbers such that the sum is 5 (here, `1` and `4`). Assign those to y and z. Now the function is defined as `aHandD[n,t,m]`, that is, to drop y and z from the list. Thus the list, or cards, becomes `2,3,5,3,7`. Now what is important here is that this is a recursive function. That is, once the numbers y and z are dropped, the function looks at the remaining arguments and tries to identify the next two numbers of which the sum is 5. This is going to be repeated until there are no such numbers. This is called a recursive function, which is the theme of Chapter 11. The rest is easy, we want the sum of the remaining numbers. So we can simply change the head of this expression to `Plus` to get the sum of the cards.

```
Apply[Plus,%]
15
```

Now we produce a list of 7 random numbers and write a little function, call it `aHand`, to put all these lines together:

```
Table[RandomInteger[{1,9}],{7}]
{5,2,7,3,1,6,3}

Apply[Sequence,%]
Sequence[5,2,7,3,1,6,3]

aHandD[%]
aHandD[5,7,1,6,3]

aHand=Module[{},
aHandD[Apply[Sequence, Table[Random[Integer, {1,
9}], {7}]]];
  Print[Apply[Plus, %]]
  ]
```

We shall see a similar approach to a problem involving matrices in Problem 12.4.

11
Recursive functions

A recursive function is a function which calls itself in its definition (true, pretty confusing!). However, recursive functions arise very naturally. This chapter studies recursive functions and how these functions are handled with ease in *Mathematica*.

Imagine two mirrors are placed (almost) parallel to each other with an apple sitting in between. Then one can see an infinite number of apples in the mirrors. This might give an impression of what a recursive function is. The classic example is Fibonacci numbers. Consider the sequence of numbers starting with 1 and 1 and continuing with the sum of the two previous numbers as the next number in the sequence. Following this rule, one obtains the sequence $1, 1, 2, 3, 5, 8, 13, 21, \cdots$. To define this sequence mathematically, one writes $F_1 = F_2 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

One can use Wolfram *Mathematica*® to define Fibonacci numbers in the exact same way recursively:

```
f[1] = 1; f[2] = 1;
f[n_] := f[n-1] + f[n-2]

f /@ Range[10]
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55}

Fibonacci /@ Range[10]
{1, 1, 2, 3, 5, 8, 13, 21, 34, 55}
```

Now, using `f`, try to compute the 50th Fibonacci number. This will take a ridiculously long time. What is the problem? The problem will show itself if you try to calculate, say, $f[5]$ by hand. By definition, $f[5] = f[4] + f[3]$ thus one needs to calculate $f[4]$ and $f[3]$. Again by definition $f[4] = f[3] + f[2]$ and $f[3] = f[2] + f[1]$. Thus in order to find the value of $f[4]$ one needs to find out $f[3]$ and $f[2]$ and for $f[3]$ one needs to calculate $f[2]$ and $f[1]$. Thus

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_11, © Springer-Verlag London Limited 2010

Mathematica is trying to calculate $f[3]$ twice unnecessarily. This shows that in order to save time, one needs to save the values of the functions in the memory. This has been done in the following code. Compare this with the above.

```
Clear[f]

f[1] = 1; f[2] = 1;
f[n_] := f[n] = f[n - 1] + f[n - 2]

f[50]
12586269025
```

Problem 11.1

Using a recursive definition, write the function

$$e(n) = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}.$$

$\Longrightarrow$ Solution.

This is Problem 6.1 in which we used `Sum` to write the function. In fact we have seen two ways to write this function so far:

```
e[n_] := 1 + Sum[1/k!, {k, 1, n}]

e[10]
9864101/3628800
```

Or use list-based programming as in Chapter 4:

```
e[n_] := 1 + Plus @@ (1/Range[n]!)

e[10]
9864101/3628800
```

Now we can write the same function but using a recursive method as follows:

```
e[1] = 1 + 1
2

e[n_] := e[n] = e[n - 1] + 1/n!
e[10]

9864101/3628800
```

Now let us calculate `e[300]` using the last function:

```
e[300]

$RecursionLimit::reclim: Recursion depth of 256 exceeded. >>

$RecursionLimit::reclim: Recursion depth of 256 exceeded. >>
```

This says *Mathematica* has a limitation on the number of recursive evaluations. By default this is 256 (*Mathematica* would circle around herself up to 256 times!). If we need to have more iterations, we shall change this by `RecursionLimit`. We will change this to 6000 and then calculate $e(5000)$ using all three definitions and will time this to see which function is fastest.

```
{$RecursionLimit = 6000}
{6000}

Timing[e[5000]][[1]]
 7.77096

Clear[e]

e[n_] := 1 + Sum[1/k!, {k, 1, n}]

Timing[e[5000]][[1]]
7.75723

Clear[e]

e[n_] := 1 + Plus @@ (1/Range[n]!)

Timing[e[5000]][[1]]
0.842002
```

Exercise 11.1

Recently Eric Rowland from Rutgers has come up with the following formula to generate prime numbers.[1] Consider the recursive function described as follows: $a(1) = 7$ and $a(n) = a(n-1) + \gcd(n, a(n-1))$ where gcd is the greatest common divisor (`GCD` function in *Mathematica*). He then proves that, for any n, $a(n) - a(n-1)$ is either 1 or a prime number. Try out this function and find some prime numbers!

Problem 11.2

Let the intertwined recursive functions s and t be defined as follows: $s(1) = 3, s(2) = 5, t(1) = 1$ and

$$\begin{aligned} s(n) &= t(n-1) - t(n-2) + 4 \\ t(n) &= t(n-1) + s(n-1). \end{aligned}$$

Check that $t(1000) = 10000$.

[1] E. Rowland, A Natural Prime-Generating Recurrence, Journal of Integer Sequences, Vol. 11 (2008), Article 08.2.8

$\Longrightarrow$ Solution.

```
s[1] = 3; s[2] = 5; t[1] = 1;

s[n_] := s[n] = t[n - 1] - t[n - 2] + 4

t[n_] := t[n] = t[n - 1] + s[n - 1]
```

Let us first check this for smaller values of n. Notice that $t(4) = t(3) + s(3)$ thus, in order to evaluate $t(4)$, one needs to evaluate $s(3)$ which is by definition $s(3) = t(2)+t(1)+4$. This shows the definitions of these functions are dependent on each other.

```
t[2]
4

t[3]
9

t[4]
16

t[100]
10000
```

Problem 11.3

Define the function f as follows

$$f(n) = \begin{cases} n-3 & \text{if } n \geq 1000 \\ f(f(n+6)) & \text{if } n < 1000. \end{cases}$$

Find the value $f(1)$.

$\Longrightarrow$ Solution.

The definition of the function consists of two parts, depending on the value of n. Recall, from Section 5.3, one can use an `If` statement to define functions of this type:

```
f[n_] := If[n >= 1000, n - 3, f[f[n + 6]]]
```

It is always a very good idea to test the function for some values that one can actually evaluate by hand, just to make sure the code is correct.

```
f[1003]
1000

f[999]
 999

f[1]
997
```

Here we use recursive programming to solve Problem 7.12.

Problem 11.4

A happy number is a number that if one squares its digits and adds them together, and then takes the result and squares its digits and adds them together again and keeps doing this process, one comes down to the number 1. Find all the happy ages, i.e., happy numbers up to 100.

$\Longrightarrow$ Solution.

```
f[1] = 1
f[4] = 4

f[n_] := f[Plus @@ ( IntegerDigits[n]^2)]

Select[Range[100], f[#] == 1 &]
{1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82,
86, 91, 94, 97, 100}
```

This does not seem to be a good definition for a happy number, as according to this definition and the above result there are only four years between 30 and retirement that a person is happy!

One can re-write many codes which have a repetitive nature in the form of recursion. Recall the Collatz function from Problem 10.2. One can write the function as follows

```
f[1]=1
f[n_Integer?EvenQ] := f[n/2]
f[n_Integer] := f[3n + 1]
```

Then if one applies `f` to any number one should get 1. (If not, then one has solved an 80-year-old conjecture in the negative!)

Exercise 11.2

Using a recursive definition, write the function

$$f(k) = x + \frac{x^3}{1 \times 2 \times 3} + \frac{x^5}{1 \times 2 \times 3 \times 4 \times 5} + \cdots + \frac{x^k}{1 \times 2 \times \cdots \times k}$$

(k is odd).

Problem 11.5

The polynomials $P_n(x, y)$ for $n = 1, 2, \cdots$ are defined by $P_1(x, y) = 1$ and

$$P_{n+1}(x, y) = (x + y - 1)(y + 1)P_n(x, y + 2) + (y - y^2)P_n(x, y).$$

Check first that $P_2(x, y) = xy + x + y - 1$. Then investigate that, for any n, $P_n(x, y) = P_n(y, x)$.

$\Longrightarrow$ Solution.

First we define the recursive function:

```
p[1, x_, y_] = 1;

p[n_, x_, y_] :=  p[n, x, y] =
(x + y - 1) (y + 1) p[n - 1, x, y + 2] + (y - y^2) p[n -1, x, y]
```

We check whether the definition is correct.

```
p[2, x, y]
y - y^2 + (1 + y) (-1 + x + y)

Simplify[%]
-1 + x + y + x y
```

The definition is correct. Of course we cannot expect *Mathematica* to be able to check $P_n(x, y) = P_n(y, x)$ for undefined n, so we check this for some instances of n.

```
Simplify[p[3, x, y] == p[3, y, x]]
True

Simplify[p[6, x, y] == p[6, y, x]]
True

Simplify[p[16, x, y] == p[16, y, x]]
True
```

12
Linear algebra

Computations with matrices are tedious jobs. One can easily define vectors and matrices in *Mathematica* and perform all the standard procedures of linear algebra using ready-to-use *Mathematica* functions. This chapter looks at these facilities.

12.1 Vectors

One of the questions asked in Chapter 3 was as follows: Given $\{x_1, x_2, \cdots, x_n\}$ and $\{y_1, y_2, \cdots, y_n\}$, how one can produce $\{x_1+y_1, x_2+y_2, \cdots, x_n+y_n\}$? The answer is if one considers lists as *vectors*, namely, $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ and $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$, where $\mathbf{x}$ and $\mathbf{y}$ are two vectors of dimension n, then the sum of vectors, $\mathbf{x}+\mathbf{y}$ is what we want. Then, as you might also guess, $\mathbf{xy}$ would produce $\{x_1y_1, x_2y_2, \cdots, x_ny_n\}$, i.e., all arithmetic which is done by lists are component-wise. However, there is another product in the setting of vectors, namely the *inner product* which is defined as

$$\mathbf{x}.\mathbf{y} = x_1y_1 + x_2y_2 + \cdots + x_ny_n.$$

The following shows how Wolfram *Mathematica*® handles these different operations.

```
xv = x# & /@ Range[5]
```

```
{x1, x2, x3, x4, x5}
```

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_12, © Springer-Verlag London Limited 2010

```
yv = y# & /@ Range[5]
```

$$\{y_1, y_2, y_3, y_4, y_5\}$$

```
xv + yv
```

$$\{x_1 + y_1, x_2 + y_2, x_3 + y_3, x_4 + y_4, x_5 + y_5\}$$

```
xv yv
```

$$\{x_1\, y_1, x_2\, y_2, x_3\, y_3, x_4\, y_4, x_5\, y_5\}$$

```
xv * yv
```

$$\{x_1\, y_1, x_2\, y_2, x_3\, y_3, x_4\, y_4, x_5\, y_5\}$$

```
xv.yv
```

$$x_1\, y_1 + x_2\, y_2 + x_3\, y_3 + x_4\, y_4 + x_5\, y_5$$

```
Norm[xv]
```

$$\sqrt{\text{Abs}[x_1]^2 + \text{Abs}[x_2]^2 + \text{Abs}[x_3]^2 + \text{Abs}[x_4]^2 + \text{Abs}[x_5]^2}$$

12.2 Matrices

It is known that matrix calculation is a tedious job. It will take well over 10 minutes to multiply

$$\begin{pmatrix} 2 & -3 & 13 & -4 & 8 \\ 12 & 1 & -18 & -4 & 2 \\ 18 & 21 & 10 & 0 & 9 \\ 8 & -12 & -4 & 0 & -3 \\ 15 & -7 & 2 & 4 & 2 \end{pmatrix} \times \begin{pmatrix} 11 & 34 & -21 & 0 & -43 \\ 12 & -33 & 9 & -12 & 7 \\ 16 & -7 & -43 & 84 & 3 \\ 4 & 9 & 12 & -1 & -54 \\ 7 & 22 & -5 & 23 & 0 \end{pmatrix}$$

only to obtain a wrong answer!

One can easily enter a matrix into *Mathematica* by choosing Insert: Table/Matrix from the menu. If we assign `A` to the first matrix and `B` to the second then

```
A.B // MatrixForm
```

$$\begin{pmatrix} 234 & 216 & -716 & 1316 & 148 \\ -146 & 509 & 473 & -1474 & -347 \\ 673 & 47 & -664 & 795 & -597 \\ -141 & 630 & -89 & -261 & -440 \\ 143 & 807 & -426 & 294 & -904 \end{pmatrix}$$

```
Det[A]
```

```
354 956
```

Note that, similar to multiplication of vectors, one should use `Dot` multiplication for multiplying matrices, i.e., `A.B`. One uses the function `MatrixForm` to obtain the result in, well, matrix form! Otherwise one gets a list of vectors. The command `Det` calculates the determinant of the matrix.

Even more impressive is how easily *Mathematica* computes the inverse of this matrix. Try

```
Inverse[A] // MatrixForm
```

$$\begin{pmatrix} -\frac{828}{88\,739} & -\frac{375}{88\,739} & \frac{3310}{88\,739} & \frac{6670}{88\,739} & -\frac{1203}{88\,739} \\ -\frac{2462}{88\,739} & -\frac{472}{88\,739} & \frac{2983}{88\,739} & \frac{113}{88\,739} & -\frac{2934}{88\,739} \\ \frac{1077}{88\,739} & -\frac{4335}{88\,739} & \frac{2768}{88\,739} & \frac{6114}{88\,739} & -\frac{3258}{88\,739} \\ -\frac{692}{12\,677} & -\frac{335}{50\,708} & -\frac{2959}{50\,708} & -\frac{10\,099}{50\,708} & \frac{4787}{25\,354} \\ \frac{6204}{88\,739} & \frac{6668}{88\,739} & -\frac{6796}{88\,739} & -\frac{20\,397}{88\,739} & \frac{12\,872}{88\,739} \end{pmatrix}$$

One can generate a matrix by using `Array` as the following example demonstrates:

```
m = Array[#1 - #2 &, {3, 4}]
```

```
{{0, -1, -2, -3}, {1, 0, -1, -2}, {2, 1, 0, -1}}
```

```
m // MatrixForm
```

$$\begin{pmatrix} 0 & -1 & -2 & -3 \\ 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 \end{pmatrix}$$

Here, `Array` acts as a nested loop (see Section 7.2). Namely, the first variable

(here `#1`) runs from 1 to 3 while the second variable (`#2`) runs from 1 to 4. This is equivalent to

```
Table[i - j, {i, 1, 3}, {j, 1, 4}]
```

```
{{0, -1, -2, -3}, {1, 0, -1, -2}, {2, 1, 0, -1}}
```

However, it is much more convenient and more readable to use `Array` when generating matrices.

Problem 12.1

Define a 3×2 matrix (a_{ij}) with entries $a_{ij} = i - j$.

$\Longrightarrow$ SOLUTION.

We will use `Array` to generate the matrix.

```
Array[#1 - #2 &, {3, 2}] // MatrixForm
```

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \\ 2 & 1 \end{pmatrix}$$

Note that in the `Array`, `#1` stands for i and `#2` for j.

A common mistake here is to use the command `MatrixForm` in the definition of a matrix. Suppose we want to define an $n \times n$ matrix (a_{ij}) with entries $a_{ij} = i - j^i$. It is very tempting to define the function as

```
m[n_] := Array[#1 - #2^#1 &, {n, n}] // MatrixForm
```

And everything looks fine and we get the matrix form of what we want.

```
m[3]
```

$$\begin{pmatrix} 0 & -1 & -2 \\ 1 & -2 & -7 \\ 2 & -5 & -24 \end{pmatrix}$$

However, let us calculate the determinant of this function.

```
Det[m[3]]
```

$$\text{Det}\left[\begin{pmatrix} 0 & -1 & -2 \\ 1 & -2 & -7 \\ 2 & -5 & -24 \end{pmatrix}\right]$$

The reason is, using `MatrixForm`, our output is no longer a two-dimensional list that *Mathematica* interprets as a matrix but a table of data.

Problem 12.2

Write a function to check that, for any n, the following identity holds:

$$\det \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ b_1 & a_1 & a_1 & \cdots & a_1 & a_1 \\ b_1 & b_2 & a_2 & \cdots & a_2 & a_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ b_1 & b_2 & b_3 & \cdots & b_n & a_n \end{pmatrix} = (a_1 - b_1)(a_2 - b_2)\cdots(a_n - b_n)$$

$\Longrightarrow$ Solution.

Here is the definition of the matrix using `Array`.

```
m[n_] := Array[
  Which[
    #1 == 1, 1,
    #1 ≤ #2, a#1-1,
    True, b#2] &,
  {n + 1, n + 1}]
```

Let's check that `m` actually produces matrices of the above form.

```
m[5] // MatrixForm
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ b_1 & a_1 & a_1 & a_1 & a_1 & a_1 \\ b_1 & b_2 & a_2 & a_2 & a_2 & a_2 \\ b_1 & b_2 & b_3 & a_3 & a_3 & a_3 \\ b_1 & b_2 & b_3 & b_4 & a_4 & a_4 \\ b_1 & b_2 & b_3 & b_4 & b_5 & a_5 \end{pmatrix}$$

```
m2[n_] := Product[a_i - b_i, {i, 1, n}]

m2[7]

(a_1 - b_1) (a_2 - b_2) (a_3 - b_3) (a_4 - b_4) (a_5 - b_5) (a_6 - b_6) (a_7 - b_7)

Simplify[Det[m[7]] == m2[7]]

True
```

Problem 12.3

Write a function to check that, for any n, the following identity holds:

$$\det \begin{pmatrix} x & a_1 & a_2 & \cdots & a_n \\ a_1 & x & a_2 & \cdots & a_n \\ a_1 & a_2 & x & \cdots & a_n \\ \vdots & \vdots & \vdots & & \vdots \\ a_1 & a_2 & a_3 & \cdots & x \end{pmatrix} = (x + a_1 + \cdots + a_n)(x - a_1) \cdots (x - a_n)$$

$\Longrightarrow$ Solution.

The solution is left to the reader this time!

Exercise 12.1

Write a function to accept a matrix A_{nn} and produce the $n^2 \times n^2$ matrix B as follows,

$$\begin{pmatrix} \begin{pmatrix} a_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{11} \end{pmatrix} & \begin{pmatrix} a_{12} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{12} \end{pmatrix} & \cdots & \begin{pmatrix} a_{1n} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{1n} \end{pmatrix} \\ \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ \begin{pmatrix} a_{n1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{n1} \end{pmatrix} & \cdots & \cdots & \begin{pmatrix} a_{nn} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{nn} \end{pmatrix} \end{pmatrix}.$$

Then show that $\det(A)^n = \det(B)$.

Exercise 12.2

Define a matrix as follows:

$$d(n) = \begin{pmatrix} 1 & 2 & \cdots & n \\ n+1 & n+2 & \cdots & 2n \\ \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & n^2 \end{pmatrix}.$$

Check that, for any $n > 2$, $\det(d(n)) = 0$.

The following is a nice problem demonstrating the use of pattern matching in *Mathematica* for solving problems.

Problem 12.4

Let A and B be 3×3 matrices. Show that $(ABA^{-1})^5 = AB^5A^{-1}$.

$\Longrightarrow$ Solution.

Let us first take the naive approach. We define two arbitrary matrices and, using *Mathematica*, we will multiply them and check whether both sides give the same result.

```
(A = Array[x#1,#2 &, {3, 3}]) // MatrixForm
```

$$\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{pmatrix}$$

```
(B = Array[y#1,#2 &, {3, 3}]) // MatrixForm
```

$$\begin{pmatrix} y_{1,1} & y_{1,2} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \\ y_{3,1} & y_{3,2} & y_{3,3} \end{pmatrix}$$

```
IA = Inverse[A];
IB = Inverse[B];
```

Now we check the equality for $n = 3$ and take the time:

```
Timing[d = A.B.IA.A.B.IA.A.B.IA; d1 = A.B.B.B.IA;
 Simplify[d == d1]]
```

```
{61.9128, True}
```

This already takes a long time. One can easily prove by induction that $(ABA^{-1})^n = AB^nA^{-1}$ for any positive integer n. For example for $n = 2$ we have $(ABA^{-1})^2 = ABA^{-1}ABA^{-1} = ABBA^{-1} = AB^2A^{-1}$. This shows a pattern here. Namely we can easily cancel A with A^{-1} if they are adjacent to each other. We introduce this to *Mathematica* and try to check the equality this way. This is very similar in nature to Problem 10.4.

```
ml[x___, y_, z_, t___] := ml[x, t] /; Simplify[y.z == IdentityMatrix[3]]

ml[r___] := Apply[Dot, {r}]

ml[] = 1;
```

```
Timing[Simplify[ml[A, B, IA, A, B, IA, A, B, IA] == A.B.B.B.IA]]
```

```
{0.005241, True}
```

♣ TIPS

– For a square matrix, `Eigenvalues` and `Eigenvectors` determine these invariants of a matrix.

Exercise 12.3

Let $b_1, b_2, b_3, b_4, \ldots$ denote the sequence defined by $b_1 = 1, b_2 = 1, b_3 = 2$ and

$$\det\begin{pmatrix} b_1 & b_1+b_2 & 1 \\ b_3 & b_3+b_4 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \det\begin{pmatrix} b_2 & b_2+b_3 & 1 \\ b_4 & b_4+b_5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$= \det\begin{pmatrix} b_3 & b_3+b_4 & 1 \\ b_5 & b_5+b_6 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \cdots = 1$$

Check that, for any given i, b_i is an integer.

13
Graphics

This chapter looks at graphics: how to use *Mathematica* to plot complex graphs. We first look at two-dimensional graphs and then concentrate on three-dimensional graphs. *Mathematica* offers similar tools in both two- and three-dimensional cases.

Graphics is one of the strongest features of Wolfram *Mathematica*®. One can use *Mathematica* to create the plot of a very complex function. We first look at two-dimensional graphs and then concentrate on three-dimensional graphs. *Mathematica* offers similar tools in both two- and three-dimensional cases.

13.1 Two-dimensional graphs

It is always very helpful to present the behavior of a function or an equation by plotting its graph. For functions with one variable, this can be done by two-dimensional graphs. Functions can come in different forms, and *Mathematica* has specific commands to handle each case. The following table shows what command is suitable for different formats of functions.

Function	Example	Graphics command
$f(x)$	$\sin(x)/x$	`Plot`
$x = f(t), y = g(t)$	$x = \sin(3t), y = \cos(4t)$	`ParametricPlot`
$f(x, y) = 0$	$x^4 - (x^2 - y^2) = 0$	`ContourPlot`
$f(x, y) \geq 0$	$x^4 + (x - 2y^2) > 0$	`RegionPlot`
$r = f(\theta)$	$r = 3\cos(5\theta)$	`PolarPlot`

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_13, © Springer-Verlag London Limited 2010

Problem 13.1

Plot the graph of the following functions:

$$f(x) = \sin(x)/x$$
$$x = \sin(3t), y = \cos(4t)$$
$$x^4 - (x^2 - y^2) = 0$$
$$x^4 + (x - 2y^2) > 0$$
$$r = 3\cos(6\theta).$$

$\Longrightarrow$ SOLUTION.

As the table in page 135 shows, to plot the graph of $\sin(x)/x$ we need to use `Plot`.

```
Plot[Sin[x]/x, {x, 0, 10 Pi}]
```

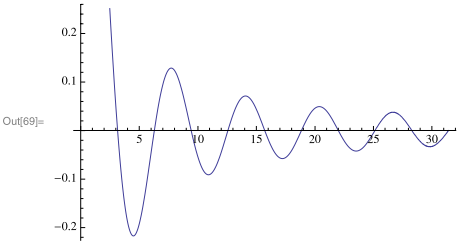


Figure 13.1 $f(x) = \sin(x)/x$

For $x = \sin(3t), y = \cos(4t)$, the command `ParametricPlot` is available.

```
ParametricPlot[{Sin[3 t], Cos[4 t]}, {t, 0, 2 Pi}]
```

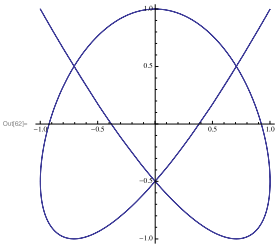


Figure 13.2 $x = \sin(3t), y = \cos(4t)$

To find the graph of all the points (x, y) which satisfy the equation $x^4 - (x^2 - y^2) = 0$, one uses `ContourPlot`.

```
ContourPlot[x^4 - (x^2 - y^2) == 0, {x, -1, 1}, {y, -1, 1}]
```

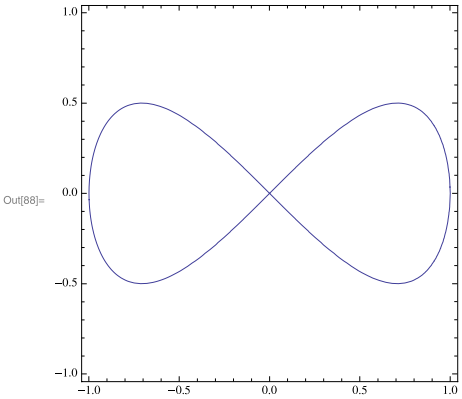


Figure 13.3 $x^4 - (x^2 - y^2) = 0$

To obtain the region, namely all the points (x, y) which satisfy the inequality $x^4 + (x - 2y^2) > 0$, we have `RegionPlot`.

```
RegionPlot[x^4 + (x - 2 y^2) >= 0, {x, -2, 2}, {y, -2, 2}]
```

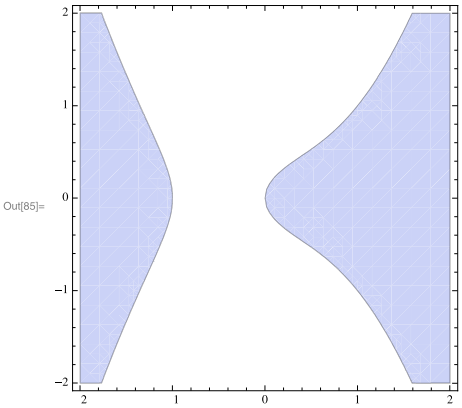


Figure 13.4 $x^4 + (x - 2y^2) > 0$

And finally to plot the polar graph of $r = 3\cos(6\theta)$, we need to use `PolarPlot`.

```
PolarPlot[3 Cos[6 t], {t, 0, 2 Pi}]
```

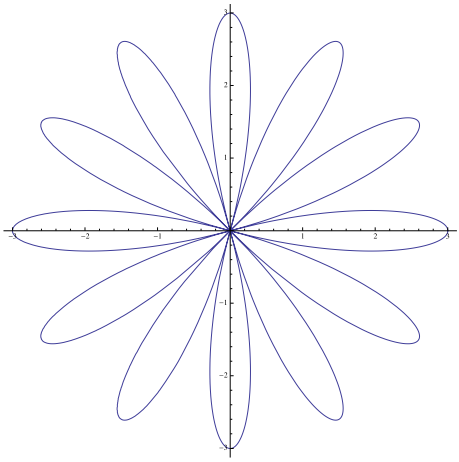


Figure 13.5 $r = 3\cos(6\theta)$

Note that in all the cases, we need to specify an interval over which we would like to plot the graph.

Problem 13.2

Show that there are 11 solutions to the equation $2\cos^2(x) - \sin(x/2) = 1$ in the interval $[0, 3\pi]$.

$\Longrightarrow$ Solution.

The best approach here is to plot the graph of $2\cos^2(x) - \sin(x/2) - 1$ and count the number of times the graph crosses the x-axis.

```
Plot[2 Cos[2 x]^2 - Sin[x/2] - 1, {x, 0, 3 Pi}]
```

From the graph (Fig. 13.6), it seems there are 11 solutions for this equation. However, there might be some concern around the region $3.1 \leq x \leq 3.2$, as it is not clear whether the graph actually touches the x-axis, namely whether there is a root there or not. So we focus on that region.

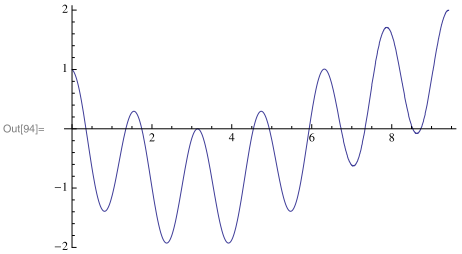


Figure 13.6 $2\cos^2(x) - \sin(x/2) - 1$

```
Plot[2 Cos[2 x]^2 - Sin[x/2] - 1, {x, 3.1, 3.2}]
```

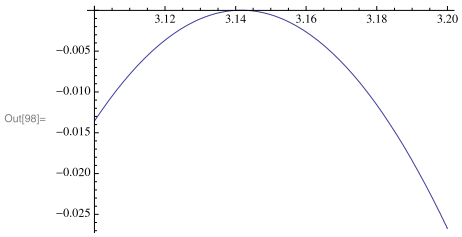


Figure 13.7 $2\cos^2(x) - \sin(x/2) - 1$ in the region $3.1 \leq x \leq 3.2$

This graph (Fig. 13.7), makes it clear that in fact there is a root in this region. We can use *Mathematica*'s ability to find roots to make sure this is the case. (For more on this see Chapter 14.)

```
FindRoot[2 Cos[2 x]^2 - Sin[x/2] - 1, {x, 3.14}]
{x -> 3.14159}
```

Problem 13.3

Observe the behavior of the graph $\sin(x) - \cos(nx)$ between 0 and π as n changes from 1 to 100.

$\Longrightarrow$ Solution.

In order to record the changes in the behavior of the graph as n runs from 1 to 100 we use `Manipulate`. The code is easy, just wrap the `Plot` around the command `Manipulate` and let `n` run from 1 to 100.

```
Manipulate[Plot[Sin[x] - Cos[n x], {x, 0, Pi}], {n, 1, 100, 1}]
```

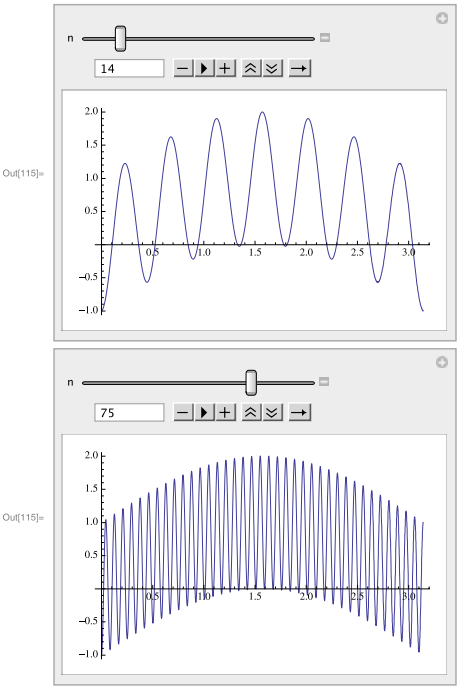


Figure 13.8 The graph of $\sin(x) - \cos(nx)$ for $n = 14$ and $n = 75$

Problem 13.4

Draw the butterfly curve, discovered by Temple H. Fay, given by

$$x(t) = \sin(t)\left(e^{\cos(t)} - 2\cos(4t) - \sin^5(t/12)\right)$$
$$x(t) = \cos(t)\left(e^{\cos(t)} - 2\cos(4t) - \sin^5(t/12)\right)$$

$\Longrightarrow$ Solution.

This is similar to the second equation in Problem 13.1. It is a parametric equation and thus calls for `ParametricPlot`

```
x[t_] := Sin[t] (E^Cos[t] - 2 Cos[4 t] - Sin[t/12]^5)

y[t_] := Cos[t] (E^Cos[t] - 2 Cos[4 t] - Sin[t/12]^5)

ParametricPlot[{x[t], y[t]}, {t, -50, 50}]
```

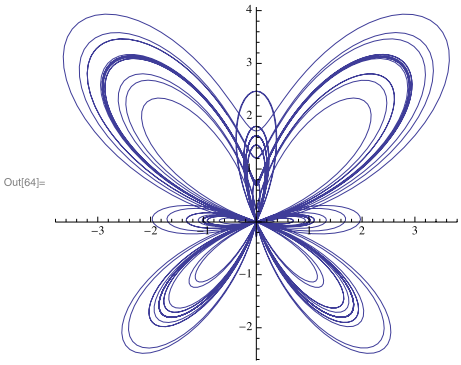


Figure 13.9 The butterfly curve

Mathematica can plot the graphs of several equations simultaneously. For this one introduces the equations into `Plot` by using a list containing all the equations.

Problem 13.5

Plot the graphs of the functions $\sin(\frac{1}{x^2-x})$ and $\frac{1.5}{x}$ in the range $[0, \pi]$.

$\Longrightarrow$ Solution.

One can plot the graph of each function separately and then use the command `Show` to show two graphs combined. However, one can also plot two graphs simultaneously (see Fig. 13.10).

```
Plot[{Sin[1/(x^2 - x)], 1.5/x}, {x, 0, Pi}]
```

One issue here is that it is not clear which graph belongs to which equation. One does have access to all aspects of graphs and can have control on all parameters which alter the graph of a function. If you type `??Plot`, you will see you can change many parameters in the graph. Here are just some samples, `AxesStyle->`, `Background->`, `FillingStyle->`, `FormatType:>`, `Frame->`,

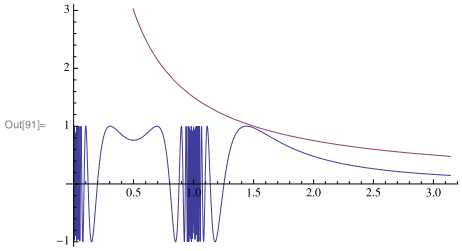


Figure 13.10 $\sin(\frac{1}{x^2-x})$ and $\frac{1.5}{x}$

`FrameLabel->`, `FrameStyle->`, `PlotStyle->`. For examples on how one can change these options, see *Mathematica* Help on Plot.

Having this, one can produce professional-looking graphs. Here is one attempt to make the above graph look more professional!

```
Plot[{Sin[1/(x^2 - x)], 1.5/x}, {x, 0, Pi}, Frame -> True,
 FrameStyle -> Thick, FrameLabel -> {x - axis, y - axis},
 PlotStyle -> {{Thick}, {Thick, Dashed}}, Background -> Gray,
 PlotLabel -> Demonstration of two graphs]
```

Out[110]=

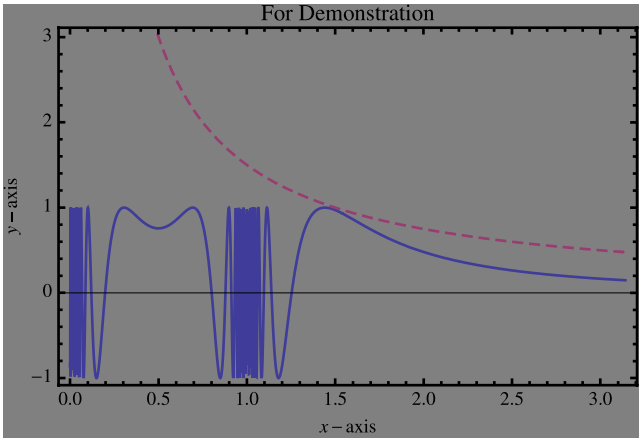


Figure 13.11 $\sin(\frac{1}{x^2-x})$ and $\frac{1.5}{x}$

Exercise 13.1

Plot the graph

$$f(x) = \sin(x) - e^{-\sum_{k=1}^{30} \frac{\cos(kx)}{k}} + e^{-\sum_{k=1}^{50} \frac{\sin(kx)}{k}}$$

for x ranging over the interval $[0, 3\pi]$. Find the maximum value that $f(x)$ takes in the interval $[0, 3\pi]$. (Hint, see `Maximize` and `NMaximize`.)

Exercise 13.2

Plot a graph of the expression

$$x(2\pi - x) \sum_{n=1}^{50} \frac{\sin(nx)}{n}$$

for $0 \leq x \leq \pi$.

Exercise 13.3

Plot the graph of

$$x(t) = 4\cos(-11t/4) + 7\cos(t)$$
$$y(t) = 4\sin(-11t/4) + 7\sin(t)$$

for $0 \leq t \leq 14\pi$.

Exercise 13.4

Plot the graph of

$$x(t) = \cos(t) + 1/2\cos(7t) + 1/3\sin(17t)$$
$$y(t) = \sin(t) + 1/2\sin(7t) + 1/3\cos(17t)$$

for $0 \leq t \leq 14\pi$.

Problem 13.6

Plot the graph of the function

$$f(x) = \begin{cases} -x, & \text{if } |x| < 1 \\ \sin(x), & \text{if } 1 \leq |x| < 2 \\ \cos(x), & \text{otherwise.} \end{cases} \tag{13.1}$$

$\Longrightarrow$ Solution.

We defined this function in Problem 5.7, using `Which` and `Piecewise`. Plotting the graph is a no-brainer, we need to plug the function $f(x)$ into the command `Plot`. Let us do so, for both definitions of the function $f(x)$.

```
f[x_] := Which[
  Abs[x] < 1, -x,
  1 <= Abs[x] < 2, Sin[x],
  True, Cos[x] ]
Plot[f[x], {x, -4, 4}]
```

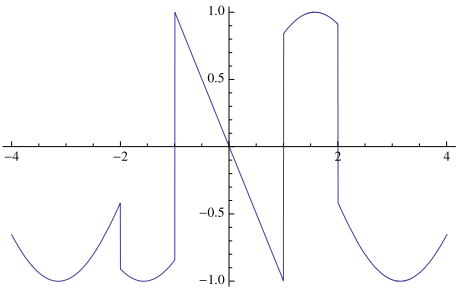


Figure 13.12 Graph of $f(x)$ using `Which`

There is also another way to define this function using the command `Piecewise`.

```
g[x_] := Piecewise[{{-x, Abs[x] < 1}, {Sin[x], 1 <= Abs[x] < 2}},
  Cos[x]]
Plot[g[x], {x, -4, 4}]
```

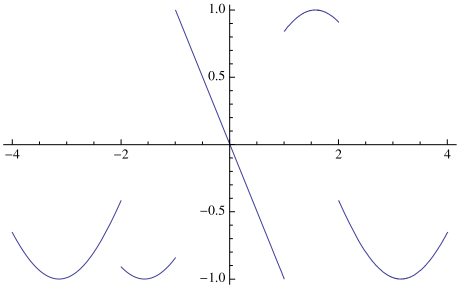


Figure 13.13 Graph of $f(x)$ using `Piecewise`

Comparing the two graphs clearly shows that the correct definition of the function is to use `Piecewise` where *Mathematica* does not interpret the function as a continuous function.

♣ TIPS

– *Mathematica* produces the graph of a function based on the number of sample points it generates. For more involved functions and higher quality graphs increase `PlotPoints` and `MaxRecursive` (see Problem 13.12).

– To arrange several plots next to each other, use `GraphicsGrid`, `GraphicsRow` and `GraphicsColumn` (see Problem 13.9 for an example on `GraphicsGrid`).

Exercise 13.5

Plot the graphs of the functions $2\exp^{-x^2}$ and $\cos(\sin(x)+\cos(x))$ between $[-\pi,\pi]$.

Exercise 13.6

Plot the graph $\mid 3x^2+xy^2-12 \mid = \mid x^2-y^2+4 \mid$.

Exercise 13.7

Plot the graph of the inequality $\mid x^2+y \mid \leq \mid y^2+x \mid$.

Exercise 13.8

Plot the graph of $4(x^2+y^2-x)^3-27(x^2+y^2)^2=0$.

Problem 13.7

Plot the graphs of $x^4-(x^2-y^2)=0$ and $y^4-(y^2-x^2)=0$. Using `Manipulate`, observe how the coefficients $0\leq a\leq 1$ and $0\leq b\leq 1$ would rescale the graph in

$$(ax)^4-((ax)^2-(ay)^2)=0,$$
$$(by)^4-((by)^2-(bx)^2)=0.$$

$\Longrightarrow$ Solution.

From the table on page 135 it is clear that one needs to use `ContourPlot` here. As in the case of `Plot`, one can feed a list of equations into `ContourPlot` to have a graph of several equations simultaneously (see Problem 13.5). Here is the code with `Manipulate`.

```
Manipulate[
 ContourPlot[{(a x)^4 - ((a x)^2 - (a y)^2) == 0,
 (b y)^4 - ((b y)^2 - (b x)^2) == 0}, {x, -1, 1}, {y, -1, 1}],
 {a, 1, 2}, {b, 1, 2}]
```

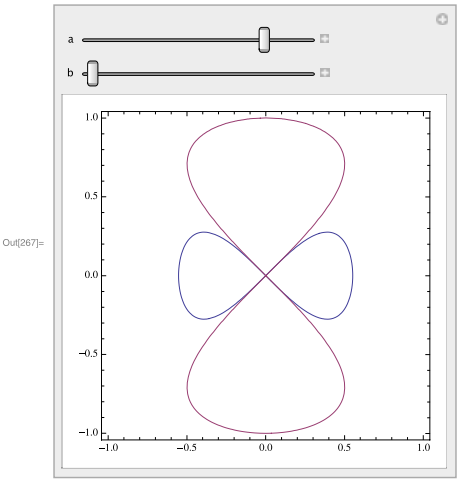


Figure 13.14 $x^4 - (x^2 - y^2) = 0$ and $y^4 - (y^2 - x^2) = 0$

Problem 13.8

Consider the first 10000 digits of $\sqrt{2}$ and present them as a "random walk" by converting them in base 4 representing 4 directions (up, down, left and right). We know that $\sqrt{2}$ is an irrational number and irrational numbers have decimal expansions that neither terminate nor become periodic. Write a code to produce this random walk. Try this code with $\sqrt{3}$, $\sqrt{6}$ and $\sqrt{13}$. Is there any comparison one can make among these numbers?

$\Longrightarrow$ Solution.

I found this problem and its slick approach in a blog on the Internet.[1] The code uses `FoldList` to push the object into different directions consecutively, thus creating a random walk. This idea is used in many similar situations (see, e.g, §2.3 in [6] and §11.2 in [2]). Let us start step by step. We first get the first 20 digits of $\sqrt{2}$ and convert them to base 4. All this can be done by `RealDigits`.

[1] http://mathgis.blogspot.com/

```
x = N[Sqrt[2], 20]
1.4142135623730950488

walk = First@RealDigits[x, 4]
{1, 1, 2, 2, 2, 0, 0, 2, 1, 3, 2, 1, 2, 1, 2, 1, 3, 3, 3, 0,
3, 2, 3, 3, 0, 3, 0, 2, 1, 0, 0, 2, 0}
```

Then when we get to zero, we move the object one step up, that is, adding $(0, 1)$ to the point; when we get to 1, we move the object one step to the right, that is adding $(1, 0)$ to the point and so on:

```
{{0, 1}, {1, 0}, {0, -1}, {-1, 0}}[[# + 1]] & /@ walk

{{1, 0}, {1, 0}, {0, -1}, {0, -1}, {0, -1}, {0, 1}, {0,
  1}, {0, -1}, {1, 0}, {-1, 0}, {0, -1}, {1, 0}, {0, -1},
  {1, 0}, {0, -1}, {1, 0}, {-1, 0}, {-1, 0}, {-1, 0}, {0, 1},
  {-1, 0}, {0, -1}, {-1, 0}, {-1, 0}, {0, 1}, {-1, 0}, {0, 1},
  {0, -1}, {1, 0}, {0, 1}, {0, 1}, {0, -1}, {0, 1}}
```

The next step is to take these movements into account consecutively. For this `FoldList` is an excellent tool to use (see Section 7.4).

```
FoldList[Plus, 0, {a, b, c}]
{0, a, a + b, a + b + c}

rn = FoldList[
  Plus, {0, 0}, {{0, 1}, {1, 0}, {0, -1}, {-1, 0}}[[# + 1]] &
  /@ walk]

{{0, 0}, {1, 0}, {2,
  0}, {2, -1}, {2, -2}, {2, -3}, {2, -2}, {2, -1}, {2, -2},
  {3, -2}, {2, -2}, {2, -3}, {3, -3}, {3, -4}, {4, -4}, {4, -5},
  {5, -5}, {4, -5}, {3, -5}, {2, -5}, {2, -4}, {1, -4}, {1, -5},
  {0, -5}, {-1, -5}, {-1, -4}, {-2, -4}, {-2, -3}, {-2, -4},
  {-1, -4}, {-1, -3}, {-1, -2}, {-1, -3}, {-1, -2}}
```

All we have to do now is to connect these points together to get a random walk.

```
Graphics[{Line[rn], PointSize[Large], Green, Point[First@rn],
Red, Point[Last@rn]}]
```

Putting all these codes together, and starting with 10000 decimal digits of $\sqrt{2}$, we have

```
x = N[Sqrt[2], 10000];
walk = First@RealDigits[x, 4];
rn = FoldList[
   Plus, {0, 0}, {{0, 1}, {1, 0}, {0, -1}, {-1, 0}}[[# + 1]] &
   /@  walk];
Graphics[{Line[rn], PointSize[Large], Green, Point[First@rn],
Red, Point[Last@rn]}]
```

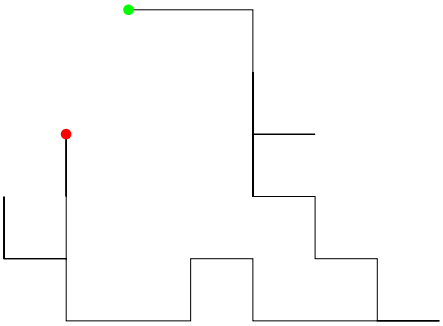


Figure 13.15 Random walks for $\sqrt{2}$

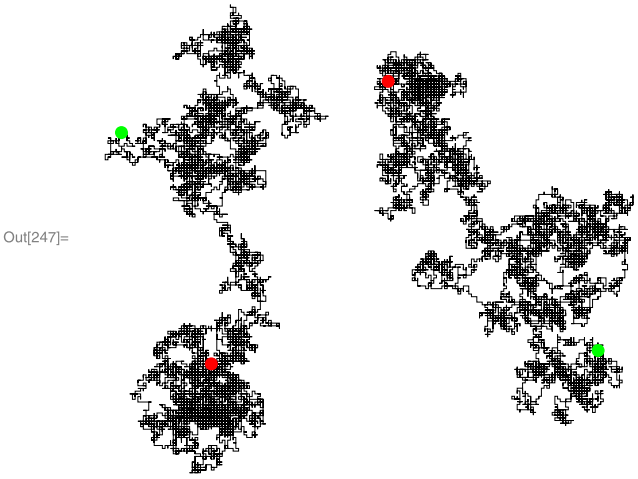


Figure 13.16 Random walks for $\sqrt{2}$ and $\sqrt{3}$

If one wants to see how the random walk actually develops, one can wrap the whole code around `Manipulate` and let the precision run from 1 to 10000 and see the fabulous result.

```
Manipulate[x = N[Sqrt[13], n];
 walk = First@RealDigits[x, 4];
 rn = FoldList[
   Plus, {0, 0}, {{0, 1}, {1, 0}, {0, -1}, {-1, 0}}[[# + 1]] &
   /@  walk];
 Graphics[{Line[rn], PointSize[Large], Green, Point[First@rn],
 Red,  Point[Last@rn]}], {n, 1, 10000, 1}]
```

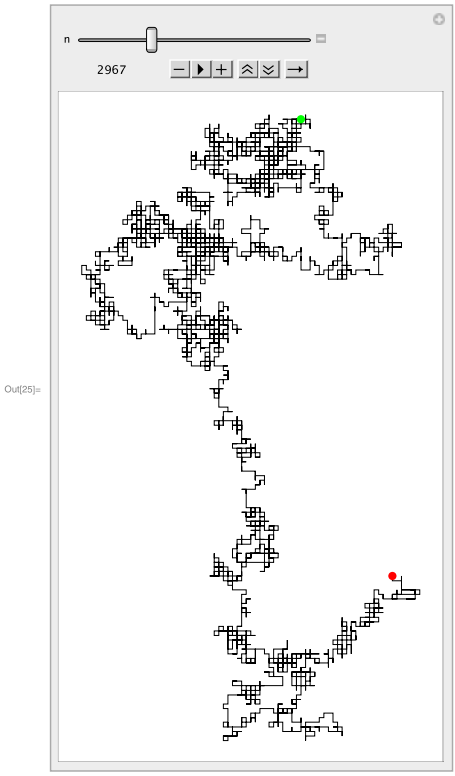


Figure 13.17 Random walk for $\sqrt{13}$

Problem 13.9

Define the Conway recursive sequence $a(1) = 1, a(2) = 1$ and

$$a(n) = a(a(n-1)) + a(n - a(n-1))$$

in *Mathematica*, and plot $a(n)/n$ when n runs from 1 to 1500.

$\Longrightarrow$ Solution.

Calculating $a(n)/n$ for $1 \leq n \leq 1500$ gives us 1500 numbers. Using `ListPlot` we can put these numbers into a figure. Recall from Chapter 11 how we handle a recursive function in *Mathematica*:

```
a[1] = a[2] = 1
a[n_] := a[n] = a[a[n - 1]] + a[n - a[n - 1]]
```

Here is the value of $a(n)/n$ for the first 20 numbers in the Conway sequence:

```
t = (a /@ Range[20])/Range[20]

{1, 1/2, 2/3, 1/2, 3/5, 2/3, 4/7, 1/2, 5/9, 3/5, 7/11,
7/12, 8/13, 4/7, 8/15, 1/2, 9/17, 5/9, 11/19, 3/5}
```

```
t = (a /@ Range[1500])/Range[1500];

?ListPlot
LisPlot[{y1,y2,...}] plots points corresponding to a list of
values, assumed to correspond to x coordinates 1, 2, ...

LisPlot[{x1,y1},{x2,y2},...}] plots a list of points with
specified x and y coordinates.

ListPlot[t]
```

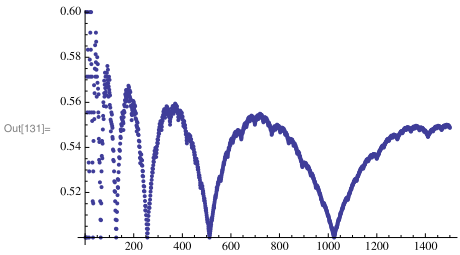


Figure 13.18 The Conway sequence

It is interesting to see that a slight change in the definition of the Conway recursive function makes the behavior of the sequence quite chaotic. Let us define a very similar recursive function to the Conway one by $b(1) = b(2) = 1$ and[2]

$$b(n) = b(b(n-1)) + b(n-1-b(n-2)).$$

Creating a similar graph as above for this function we have

```
t1 = (b /@ Range[1500])/Range[1500];

GraphicsGrid[{{ListPlot[t], ListPlot[t1]}}]
```

Notice how we have used `GraphicsGrid` to put two plots in one row (see Fig. 13.19).

[2] This has been studied by K. Pinn, A Chaotic Cousin Of Conway's Recursive Sequence, available at arXiv.org.

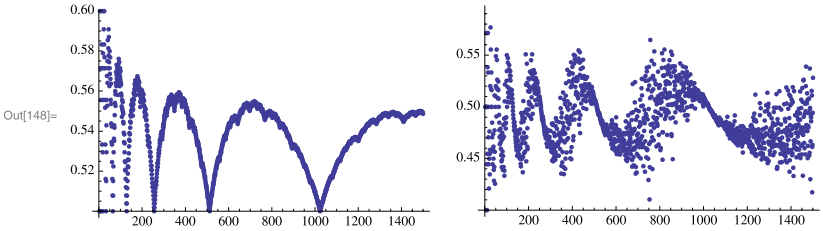


Figure 13.19 The Conway sequence and its cousin

Problem 13.10

Recall from Problem 8.2 that to get the Thue–Morse sequence, one starts with 0 and then repeatedly replaces 0 with 01 and 1 with 10. Create the 15th number in the Thue–Morse sequence. Then, based on this number, create a graph as follows: starting from $(0, 0)$, move ahead by one unit if you encounter 1 in the Thue–Morse sequence and rotate counterclockwise by an angle of $\pi/3$ if you encounter 0. (The resulting curve converges to the Koch snowflake, a fractal curve of infinite length containing a finite area.[3])

$\Longrightarrow$ Solution.

We first create the 15th Thue–Morse number. See Problem 8.2, where we have written this code. Note that this is a huge number and it might take some time until *Mathematica* produces this, and we are not going to print the output. To create the graph, we will use a similar technique to that in Problem 13.8.

```
Flatten[ReplaceRepeated[{0}, {0 -> {0, 1}, 1 -> {1, 0}},
  MaxIterations -> 4]]

ReplaceRepeated::rrlim: Exiting after {0} scanned 4 times. >>

{0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0}

Flatten[ReplaceRepeated[{0}, {0 -> {0, 1}, 1 -> {1, 0}},
   MaxIterations -> 4]] /. {1 -> -Pi/3}
```

[3] See, for example, the Thue–Morse sequence in Wikipedia.

```
ReplaceRepeated::rrlim: Exiting after {0} scanned 4 times. >>

{0, -Pi/3, -Pi/3, 0, -Pi/3, 0, 0, -Pi/3, -Pi/3, 0, 0,
-Pi/3, 0, -Pi/3, -Pi/3, 0}

Accumulate[
 Flatten[ReplaceRepeated[{0}, {0 -> {0, 1}, 1 -> {1, 0}},
    MaxIterations -> 4]] /. {1 -> -Pi/3}]

ReplaceRepeated::rrlim: Exiting after {0} scanned 4 times. >>

{0, -Pi/3, -2 Pi/3, -2 Pi/3, -Pi, -Pi, -Pi, -4Pi/3, -5Pi/3),
-5Pi/3, -5Pi/3, -2Pi, -2Pi, -7Pi/3,
-8Pi/3, -8Pi/3}

re = {Sin[#], Cos[#]} & /@
   Accumulate[
    Flatten[ReplaceRepeated[{0}, {0 -> {0, 1}, 1 -> {1, 0}},
       MaxIterations -> 15]] /. {1 -> -Pi/3}];

ReplaceRepeated::rrlim: Exiting after {0} scanned 15 times. >>

re1 = FoldList[Plus, {0, 0}, re];

Graphics[Line[re1]]
```

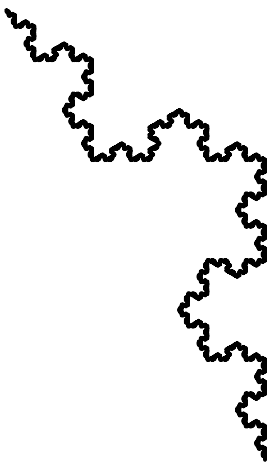


Figure 13.20 Thue–Morse fractal

13.2 Three-dimensional graphs

As in Section 13.1 with two-dimesional graphics, *Mathematica* provides several commands to handle three-dimensional graphics. A table similar to the one on page 135 can be drawn to show which command should be used in different circumstances.

Function	Example	Graphics command
$f(x, y)$	$\sin(x^2 + y^2)e^{-x^2}$	`Plot3D`
$x = f(t), y = g(t),$ $z = h(t)$	$x = \sin(3t), y = \cos(4t),$ $z = \sin(5t)$	`ParametricPlot3D`
$f(x, y, z) = 0$	$6x^2 - 2x^4 - y^2z^2 = 0$	`ContourPlot3D`
$f(x, y, z) \leq 0$	$x^4 + (x - 2y^2) > 0$	`RegionPlot3D`

Problem 13.11

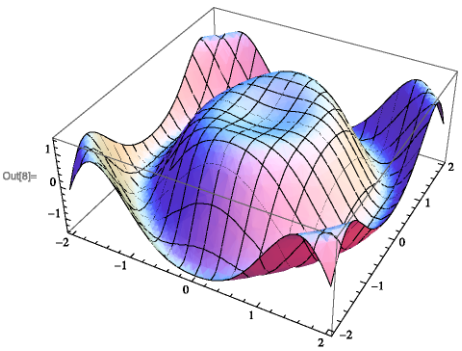
Plot the graph of the "cowboy hat" equation

$$\sin(x^2 + y^2)e^{-x^2} + \cos(x^2 + y^2)$$

as both x and y range from -2 to 2.

$\Longrightarrow$ Solution. We first translate the equation into *Mathematica* and then, using `Plot3D`, will plot the graph.

```
Sin[x^2 + y^2] Exp[-x^2] + Cos[x^2 + y^2]

Plot3D[q[x, y], {x, -2, 2}, {y, -2, 2}, PlotPoints -> 50]
```



Problem 13.12

Produce the graph of $f(x, y) = xy \sin(x^2) \cos(y^2)$ when $-2\pi \leq x \leq 0$ and $-2\pi \leq y \leq 0$. Using `PlotPoints` produce the graph with different accuracy.

$\Longrightarrow$ Solution.

Here we use `PlotPoints` to force *Mathematica* to produce less or more sample points than it usually does. One can see, if we ask *Mathematica* to produce the graph based on only 5 sample points, we get a crude sort of a graph. Changing the `PlotPoints` to 50, we will get a high quality graph.

```
Plot3D[x y Sin[x^2] Cos[y^2], {x, -2 Pi, 0}, {y, -2 Pi, 0},
 PlotRange -> All, PlotPoints -> 5]
```

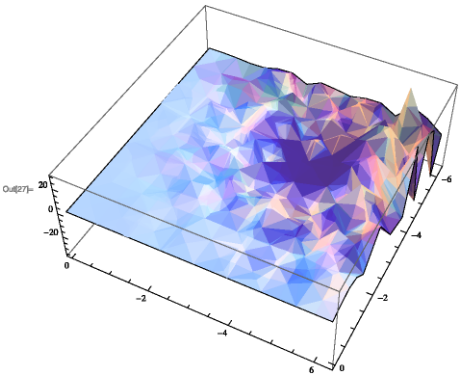


Figure 13.21 $xy \sin(x^2) \cos(y^2)$ with 5 sample points

```
Plot3D[x y Sin[x^2] Cos[y^2], {x, -2 Pi, 0}, {y, -2 Pi, 0},
 PlotRange -> All, PlotPoints -> 50]
```

One nice experiment is to observe how a graph gets more accurate as the number of `PlotPoints` increases. One way to do so is to use `Manipulate` and let `PlotPoints->i` and change `i` from, say, 5 to 30. However, as it takes a long time for each of these graphs to be produced, one does not get a smooth animation. One solution is to produce several instances and show them one after the other using `ListAnimate`. The following code does just that: we produce several "frames" of the graph, using `Table`, which changes the `PlotPoints`, and then show these frames one after the other using `ListAnimate`.

```
ListAnimate[
 Table[Plot3D[x y Sin[x^2] Cos[y^2], {x, -2 Pi, 0}, {y, -2 Pi, 0},
   PlotRange -> All, PlotPoints -> i], {i, 5, 30, 3}]]
```

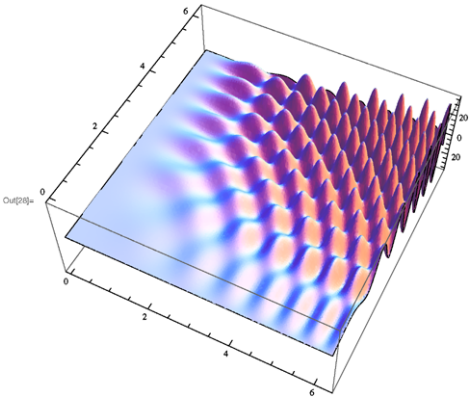


Figure 13.22 $xy\sin(x^2)\cos(y^2)$ with 50 sample points

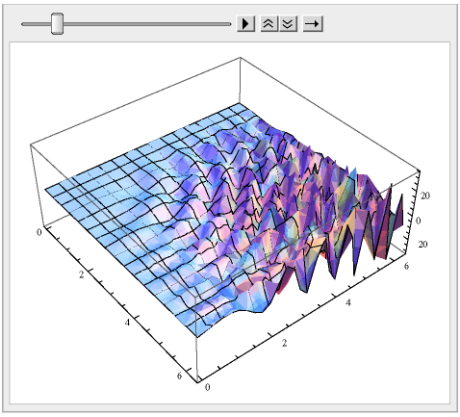


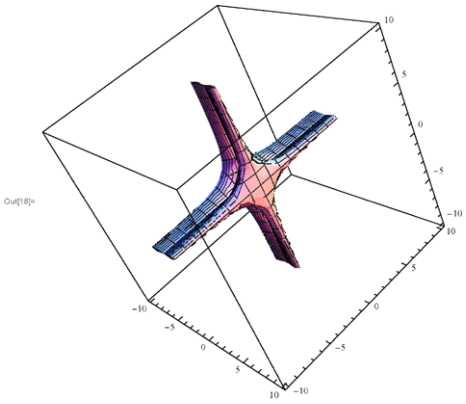
Figure 13.23 $xy\sin(x^2)\cos(y^2)$ using `ListAnimate`

Problem 13.13

Plot the graph of $6x^2 - 2x^4 - y^2z^2 = 0$.

$\Longrightarrow$ Solution.

```
ContourPlot3D[6 x^2 - 2 x^4 - y^2 z^2 == 0, {x, -10, 10},
{y, -10, 10}, {z, -10,  10}]
```



Exercise 13.9

Plot the graphs of the following.[4]

Calyx	$x^2 + y^2z^3 = z^4$
Durchblick	$x^3y + xz^3 + y^3z + z^3 + 5z = 0$
Seepferdchen	$(x^2 - y^3)^2 = (x + y^2)z^3$
Geisha	$x^2yz + x^2z^2 = y^3z + y^3$
Schneeflocke	$x^3 + y^2z^3 + yz^4 = 0$

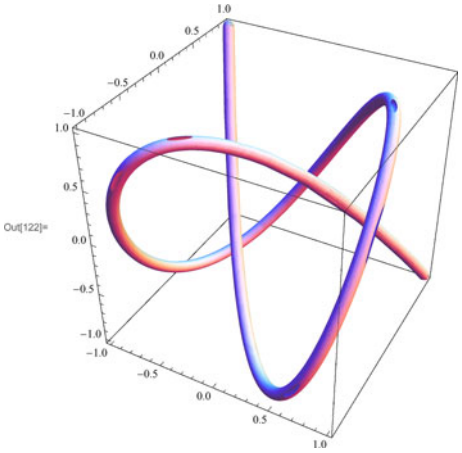
Problem 13.14

Plot the graph of $x = \sin(3t), y = \cos(4t), z = \sin(5t)$, for $-\pi \leq t \leq \pi$. Then create a dynamic setting and plot the graph of $x = \sin(nt), y = \cos(mt), z = \sin(5t)$ for $1 \leq n, m \leq 10$.

$\Longrightarrow$ Solution.

```
ParametricPlot3D[{Sin[3 t], Cos[4 t], Sin[5 t]}, {t, -Pi, Pi},
  PlotStyle -> Tube[0.05]]
```

[4] A nice gallery of these and much more can be found in the Herwig Hauser homepage http://www.freigeist.cc/gallery.html




```
Manipulate[
 ParametricPlot3D[{Sin[n t], Cos[4 m t], Sin[5 t]}, {t, -Pi, Pi},
  PlotStyle -> Tube[0.05]], {n, 1, 10, 1}, {m, 1, 10, 1}]
```

14
Calculus and equations

Mathematica comes with several powerful commands to solve different kinds of equations. Doing calculus is also one of the strong features of this software. This chapter gives a brief account of what is available here.

14.1 Solving equations

Solving equations and finding roots for different types of equations and relations are one of the main endeavors of mathematics. For polynomials with one variable, i.e., of the form $a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, it has been proved that there is no formula for finding the roots when $n \geq 5$ (in fact, when $n = 3$ or 4, the formulas are not that pretty!). This forces us to find numerical ways to estimate the roots of the equations. Using Wolfram *Mathematica*® we have several commands at our disposal. There are different kinds of equations and they require different commands to find the roots. The following table shows what command is suitable for different formats of equations.

Example	Commands to solve an equation
$x^4 - 3x^3 + 2x + 10 = 0$	`Solve`
$x^6 - 4x^3 + 12x + 10 = 0$	`NSolve`
$x^3 - 3x^2 + 5 < 0$	`Reduce`
$x^{x-10} = x^2$ and $x \in \mathbb{N}$	`FindInstance`
$\sin(x) = x - 1$	`FindRoot`

Although all the examples in the above table are equations with one variable, *Mathematica* can also handle equations with more than one variable. The

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_14, © Springer-Verlag London Limited 2010

point is, in order to find roots for the equations, one needs to experiment with the above commands to find which one would produce the solutions to the equation. In the problems below we consider several situations to demonstrate how one works with these commands.

Problem 14.1

Find the roots of the following equations:

$$x^4 - 3x^3 + 2x + 10 = 0,$$
$$x^6 - 4x^3 + 12x + 10 = 0,$$
$$x^3 - 3x^2 + 5 < 0,$$
$$x^{x-10} = x^2 \text{ and } x \in \mathbb{N},$$
$$\sin(x) = x - 1.$$

$\Longrightarrow$ Solution.

To solve the first equation, one can use `Solve`, as this is a polynomial of degree 4, thus there is an algebraic method to get all the solutions. We then use `N` to get the numerical value of these solutions.

```
N[Solve[x^4 - 3 x^3 + 2 x + 10 == 0, x]]

{{x -> -0.822108 - 1.00134 I}, {x -> -0.822108 +
    1.00134 I}, {x -> 2.32211 - 0.75191 I}, {x ->
   2.32211 + 0.75191 I}}
```

For the second equation, `Solve` is not going to give us any answer. However, using `NSolve`, we get the following roots:

```
NSolve[x^6 - 4 x^3 + 12 x + 10 == 0, x]

{{x -> -0.929435 - 0.361625 I}, {x -> -0.929435 +
    0.361625 I}, {x -> -0.62568 - 1.72428 I}, {x -> -0.62568 +
    1.72428 I}, {x -> 1.55512 - 0.754856 I}, {x ->
   1.55512 + 0.754856 I}}
```

For the third inequality, `Reduce` proved to be the right tool:

```
Reduce[x^3 - 3 x^2 + 5 < 0, x]
x < Root[5 - 3 #1^2 + #1^3 &, 1]

N[%]
x < -1.1038
```

For the fourth equation, we have:

```
FindInstance[x^(x - 10) == x^2, x, Integers]
{{x -> 12}}
```

In order to find solutions for the equation $\sin(x) = x - 1$, as this is not an algebraic equation, there is no hope of being able to get any meaningful answer using any of the commands above as the following shows:

```
Solve[Sin[x] == x - 1, x]

During evaluation of In[200]:= Solve::tdep: The equations appear
to involve the variables to be solved for in an essentially
non-algebraic way. >>
```

One gets the same message using `NSolve` and `Reduce`. The way forward is to plot the graph of this equation and see where the curve crosses the x-axis:

```
Plot[Sin[x] - x + 1, {x, -2 Pi, 2 Pi}]
```

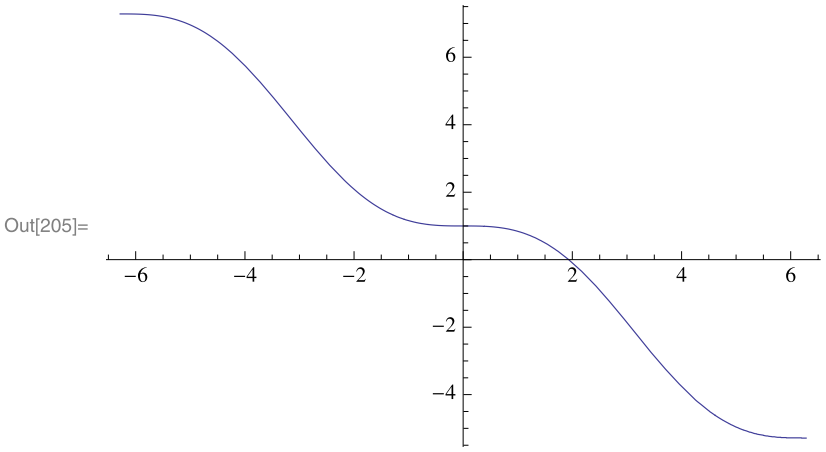


Figure 14.1 Plotting the graph and using `FindRoot`

From the graph it is clear that this equation has a root somewhere close to 2. Now using `FindRoot` and helping *Mathematica* a little, that is, giving her a point close to the root, she can find the exact root for us:

```
FindRoot[Sin[x] == x - 1, {x, 2}]
{x -> 1.93456}
```

As we mentioned, all the commands in the above table can also handle equations with more than one variable.

Problem 14.2

Find all pairs of real numbers (x, y) satisfying the system of equations

$$2 - x^3 = y$$
$$2 - y^3 = x + \sin(y).$$

$\Longrightarrow$ Solution.

Let us first plot the graphs of these two functions in one figure:

```
ContourPlot[{2 - x^3 == y, 2 - y^3 == x + Sin[y]}, {x, -100,
  100}, {y, -100, 100}]
```

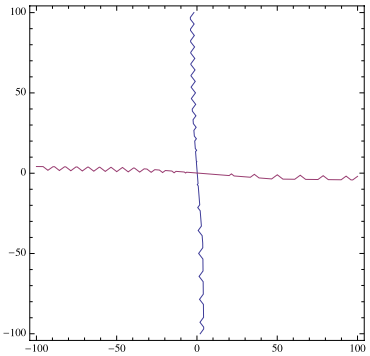


Figure 14.2 Graph of $2 - x^3 = y$ and $2 - y^3 = x + \sin(y)$

The figure shows that these two graphs intersect each other in just one point. Let us concentrate on a smaller interval:

```
ContourPlot[{2 - x^3 == y, 2 - y^3 == x + Sin[y]}, {x, -10,
  10}, {y, -10, 10}]
```

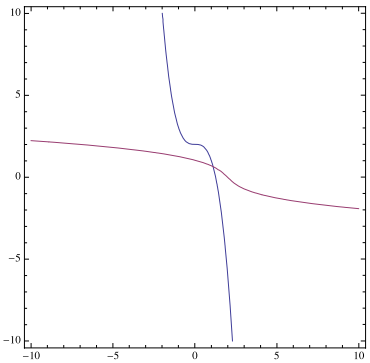


Figure 14.3 Plotting the graphs and using `FindRoot`

So we can easily find this point by using `FindRoot` and giving a point close to this root:

```
FindRoot[{2 - x^3 == y, 2 - y^3 == x + Sin[y]}, {x, 2}, {y, 1}]
{x -> 1.10294, y -> 0.658304}
```

Problem 14.3

Find two positive integers x and y such that the equation $f(x,y) = 2xy^4 + x^2y^3 - 2x^3y^2 - y^5 - x^4y + 2y$ gives the 10th Fibonacci number. (For your information: It is an extraordinary theorem which states that the Fibonacci numbers are precisely the positive values of this equation where x and y are integers.)

$\Longrightarrow$ Solution.

As we are looking for a sample solution of this equation, `FindInstance` is the right command:

```
f[x_, y_] := 2 x y^4 + x^2 y^3 - 2 x^3 y^2 - y^5 - x^4 y + 2 y

FindInstance[f[x, y] == Fibonacci[10], {x, y}, Integers]
{{x -> -89, y -> 55}}
```

The x and y we got are not positive. We try to get more sample solutions:

```
FindInstance[f[x, y] == Fibonacci[10], {x, y}, Integers, 2]
{{x -> -89, y -> 55}, {x -> 34, y -> 55}}
```

Problem 14.4

Let A_n be the $n \times n$ matrix with (i,j)-th entry equal to $x^{|i-j-ij|+ij}$. For $n = 3$ and 4 find exactly all values of x for which the determinant of A_n is zero. Find all the distinct values of x for which the determinant of A_6 is zero. Observe that there are 21 of them.

$\Longrightarrow$ Solution.

The only challenge here is to define this matrix. For this see Chapter 12. This done, we start with `Solve` to see whether we get exact roots of the equation produced by the determinant of A_3.

```
A[n_] := Array[x^(Abs[#1 - #2 - #1 #2] + #1 #2) &, {n, n}]

Det[A[3]]
-x^20 + 2 x^22 - 2 x^26 + x^28
```

```
Solve[Det[A[3]] == 0]
{{x -> -1}, {x -> -1}, {x -> -1}, {x -> 0}, {x -> 0}, {x ->
   0}, {x -> 0}, {x -> 0}, {x -> 0}, {x -> 0}, {x -> 0}, {x ->
   0}, {x -> 0}, {x -> 0}, {x -> 0}, {x -> 0}, {x -> 0}, {x ->
   0}, {x -> 0}, {x -> 0}, {x -> 0}, {x -> 0}, {x ->
   0}, {x -> -I}, {x -> I}, {x -> 1}, {x -> 1}, {x -> 1}}
```

This shows the command `Solve` is successful in finding the roots. So we stick to it. Also the above result shows there are many repeated roots. As we are interested in distinct roots, we get rid of the repetition by using `Union`.

```
Union[Solve[Det[A[4]] == 0]]
{{x -> -1}, {x -> 0}, {x -> -I}, {x -> I}, {x ->
   1}, {x -> -(-1)^(1/3)}, {x -> (-1)^(
   1/3)}, {x -> -(-1)^(2/3)}, {x -> (-1)^(2/3)}}
```

For the last part of the problem:

```
Length[Union[Solve[Det[A[6]] == 0]]]
21
```

♣ TIPS

– Using `NSolve`, one can ask *Mathematica* for more precision when producing the answer. `NSolve[equations,var,n]` would solve the equations numerically to `n` significant digits.

– `Solve` produces the answers as rules, whereas `Reduce` gives them as a Boolean statement. Plus `Reduce` will describe *all* possible solutions.

```
Solve[a x + b == 0, x]
{{x -> -(b/a)}}

Reduce[a x + b == 0, x]
(b == 0 && a == 0) || (a != 0 && x == -(b/a))
```

– `FindRoot[Equation==0,x,x0]` uses Newton's method to solve the equation. This means, if the derivative of the `Equation` can't be computed, Newton's method fails and so this command would not give any answer.

– `FindRoot[Equation==0,x,x0,x1]` uses the secant method to solve the equation. Thus, if the Newton method is not successful, one can try this approach.

14.2 Calculus

Two important machineries in calculus are derivations and integrations. We assume the reader is familiar with calculus. In the problems below we showcase some of the abilities of *Mathematica* in this area.

Example	Commands to use
$\frac{\partial f}{\partial x}$	`D[f,x]`
$\frac{\partial^2 f}{\partial x \partial y}$	`D[f,x,y]`
$\int f(x)dx$	`Integrate[f,x]`
$\int_a^b f(x)dx$	`Integrate[f,{x,a,b}]` or `NIntegrate`
$\int_c^d \int_a^b f(x,y)dxdy$	`Integrate[f[x,y],{y,c,d},{x,a,b}]`

Problem 14.5

Evaluate the following:

$$\frac{\partial f}{\partial x}, \text{ when } f = \sin(x)/x,$$
$$\frac{\partial^2 f}{\partial x^2}, \text{ when } f = \sin(x)/x,$$
$$\frac{\partial^3 f}{\partial x^2 \partial y}, \text{ when } f = e^{xy},$$
$$\int \big(\cos(x)/x - \sin(x)/x^2\big)dx$$
$$\int_{-1}^{1}\int_{-1}^{1} \cos(x^2+y^2+xy)dxdy.$$

$\Longrightarrow$ Solution.

All we need to do is to translate these into *Mathematica* according to the table above:

```
D[Sin[x]/x, x]
Cos[x]/x - Sin[x]/x^2

D[Sin[x]/x, x, x]
-((2 Cos[x])/x^2) + (2 Sin[x])/x^3 - Sin[x]/x

D[E^(x  y), x, x, y]
2 E^(x y) y + E^(x y) x y^2

Integrate[Cos[x]/x - Sin[x]/x^2, x]
Sin[x]/x
```

To evaluate $\int_{-1}^{1}\int_{-1}^{1} \cos(x^2+y^2+xy)dxdy$ we can also use `Integrate`, so *Mathematica* would come up with the precise answer (and it will come up with the precise answer). However, it takes some time:

```
Timing[Integrate[Cos[x^2 + y^2 + x y],
{y, -1, 1}, {x, -1, 1}]][[1]]
21.5496
```

However, if we use `NIntegrate`, that is, ask *Mathematica* to approach this calculation numerically, we get the answer in no time.

```
NIntegrate[Cos[x^2 + y^2 + x y], {y, -1, 1}, {x, -1, 1}]
2.81372
```

Problem 14.6

Consider $f(x, y) = \sin(x + y)\cos(x^2 - y) - \sin(y)$ and generate the graphs of $\frac{\partial^2 f}{\partial x \partial y}$ over the rectangle $-\pi \leq x \leq \pi$ and $-\pi \leq y \leq \pi$. Find the maximum of the function $\frac{\partial^2 f}{\partial x \partial y}$ in this area.

$\Longrightarrow$ Solution.

We define the function $f(x, y)$ and calculate its second derivative with respect to x and y:

```
f[x_, y_] := Sin[x + y] Cos[x^2 - y] - Sin[y]

s = D[f[x, y], x, y]
Cos[x + y] Sin[x^2 - y] - 2 x Cos[x + y] Sin[x^2 - y] -
 Cos[x^2 - y] Sin[x + y] + 2 x Cos[x^2 - y] Sin[x + y]
```

All we need to do is to plug this equation into `Plot3D` and plot the graph in the interval asked in the problem.

```
Plot3D[s, {x, -Pi, Pi}, {y, -Pi, Pi}]
```

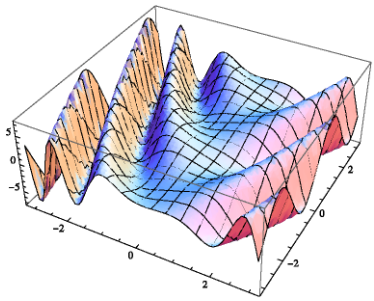


Figure 14.4 Graph of $\frac{\partial^2 f}{\partial x \partial y}$

In order to find the maximum of this function, two commands are available, `Maximize`, and its numerical version `NMaximize` which will be much faster and

will give you a numerical approximation of the result. There are also two commands for finding the minimum of a function, namely `Minimize` and `NMinimize`.

```
NMaximize[{s, -Pi <= x <= Pi, -Pi <= y <= Pi}, {x, y}]
{4.47747, {x -> 2.74964, y -> 3.14159}}
```

Problem 14.7

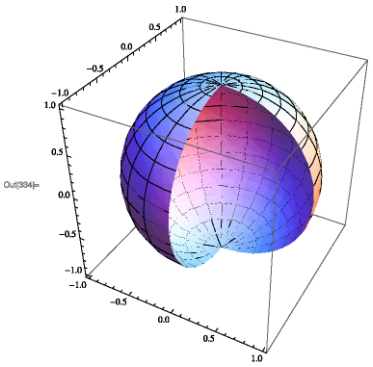
Consider the following functions of two variables $\mathbf{x}(u,v) = \sin(v)\cos(u)$, $\mathbf{y}(u,v) = \sin(v)\sin(u)$ and $\mathbf{z}(u,v) = \cos(v)$. Generate the surface $(\mathbf{x},\mathbf{y},\mathbf{z})$ when $0 \leq u \leq 3\pi/2$ and $0 \leq v \leq \pi$. Now consider $\mathbf{x_1}(u,v) = \frac{-3}{8}\cos(v)\sin(\frac{4u}{3})$, $\mathbf{y_1}(u,v) = \frac{3}{8}\cos(\frac{4u}{3})\cos(v)$ and $\mathbf{z_1}(u,v) = \frac{\sin(v)}{2}$. Generate the surface

$$(\frac{d\mathbf{x_1}}{dvdu}, \frac{d\mathbf{y_1}}{dudv}, \frac{d\mathbf{z_1}}{dv})$$

when $0 \leq u \leq 3\pi/2$ and $0 \leq v \leq \pi$. Finally, superimpose these two images.

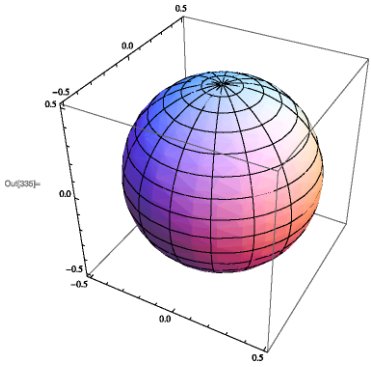
$\Longrightarrow$ Solution.

```
x[u_, v_] := Sin[v] Cos[u]
y[u_, v_] := Sin[v] Sin[u]
z[u_, v_] := Cos[v]

ParametricPlot3D[{x[u, v], y[u, v], z[u, v]}, {u, 0, 3 Pi/2},
{v, 0, Pi}]
```

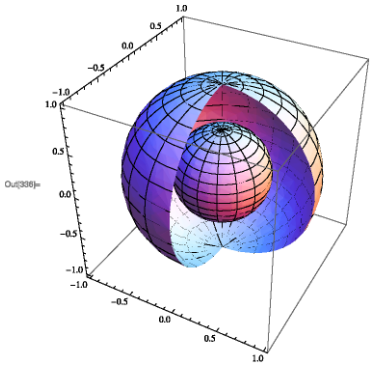


```
x1[u_, v_] := -3/8 Cos[v] Sin[4 u/3]
y1[u_, v_] := 3/8 Cos[4 u/3] Cos[v]
z1[u_, v_] := Sin[v]/2
```

```
x2 = D[x1[u, v], u, v]
1/2 Cos[(4 u)/3] Sin[v]

y2 = D[y1[u, v], v, u]
1/2 Sin[(4 u)/3] Sin[v]

z2 = D[z1[u, v], v]
Cos[v]/2

ParametricPlot3D[{x2, y2, z2}, {u, 0, 3 Pi/2}, {v, 0, Pi}]
```




```
Show[Out[334], Out[335]]
```

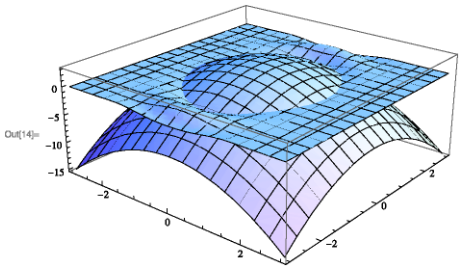


Problem 14.8

Consider two surfaces $q(x, y) = \cos(x^2 + y^2) \exp^{-x^2}$ and $w(x, y) = 3 - x^2 - y^2$. Plot these functions and show the result from side and bottom view. Find the volume of the region between the graphs on the domain $[-1, 1] \times [-1, 1]$.

$\Longrightarrow$ SOLUTION.

```
q[x_, y_] := Cos[x^2 + y^2] E^(-x^2)

w[x_, y_] := 3 - x^2 - y^2

Plot3D[{q[x, y], w[x, y]}, {x, -3, 3}, {y, -3, 3}]
```




```
NIntegrate[w[x, y] - q[x, y], {x, -1, 1}, {y, -1, 1}]
7.02707
```

15
Solutions to the Exercises

This chapter provides solutions to the selected exercises.

Exercise 1.1

In order to get the correct result, one needs to group the items in the right order. Another approach is to use the *Mathematica* palettes and type the expression as it looks into the Front end.

```
Sqrt[64^(1/3)*(2^2 + (1/2)^2) - 1]
4
```

Exercise 1.2

```
PrimeQ[123456789098765432111]
True
```

Exercise 1.3

We check this for $n = 5$.

```
n = 5;
LCM[1, 2, 3, 4, 5]
60

2^(5 - 1)
16
```

Exercise 1.4

One can enter 32! straight into *Mathematica* with confidence.

```
32!
263130836933693530167218012160000000
```

Clearly, the number ends with 7 zeros.

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3_15, © Springer-Verlag London Limited 2010

Exercise 1.5

```
Factor[(1 + x)^30 + (1 - x)^30]
2 (1 + x^2) (1 + 14 x^2 + x^4) (1 + 44 x^2 + 166 x^4 + 44 x^6 +
   x^8) (1 + 376 x^2 + 4380 x^4 + 15944 x^6 + 24134 x^8 +
   15944 x^10 + 4380 x^12 + 376 x^14 + x^16)
```

Exercise 1.6

```
Together[1/(1 + x) + 1/(1 + (1/(1 + x)))]
(3 + 3 x + x^2)/((1 + x) (2 + x))

Apart[%]
1 + 1/(1 + x) - 1/(2 + x)
```

Exercise 1.7

```
(1 + Sin[x] - Cos[x])/(1 + Sin[x] + Cos[x]) == Tan[x/2]
(1 - Cos[x] + Sin[x])/(1 + Cos[x] + Sin[x]) == Tan[x/2]

Simplify[(1 + Sin[x] - Cos[x])/(1 + Sin[x] + Cos[x]) ==
  Tan[x/2]]
True
```

Exercise 2.1

```
f[x_] := Sqrt[1 + x]

f[f[f[f[x]]]]
Sqrt[1 + Sqrt[1 + Sqrt[1 + Sqrt[1 + x]]]]
```

Exercise 3.1

```
Clear[f]

f[n_] := n^6 + 1091

RandomInteger[{1, 3095}]
590

Table[PrimeQ[f[RandomInteger[{1, 3095}]]], {6}]
{False, False, False, False, False, False}
```

Exercise 3.2 Among the first 450 Fibonacci numbers, the number of odd Fibonnaci numbers is:

```
Length[Select[Fibonacci[Range[450]], OddQ]]
300
```

Thus the number of even Fibonacci numbers is 150, which is half of the number of odd ones.

Exercise 3.3 We first define A as a function which depends on m.

```
f[m_] := ((m + 3)^3 + 1)/(3 m)

fQ[m_] := IntegerQ[((m + 3)^3 + 1)/(3 m)]

Select[Range[500], fQ]
{2, 14}

f /@ %
{21, 117}
```

This shows that, for two integers m less than 500, $f(m)$ is an integer and in both cases they are odd numbers.

There are other ways to approach this problem using `Count` and `Cases`. These approaches will be discussed in Chapter 9.

```
Cases[f[Range[500]], _Integer]
{21, 117}

Count[f[Range[500]], _Integer]
2
```

Exercise 3.4

```
Length[Select[Range[20000], Divisible[#^2 + (# + 1)^2, 1997] &]]
20

Length[Select[Range[20000], Divisible[#^2 + (# + 1)^2, 2009] &]]
0
```

The following approach uses pattern matching which will be discussed in Chapter 9.

```
Count[(Range[20000]^2 + (Range[20000] + 1)^2)/1997, _Integer]
20
```

Exercise 3.5

```
Select[Range[2, 200], Divisible[(# - 1)! + 1, #] &]
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137,
139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199}

Length[%]
46
```

Here is another way to write the code:

```
Length[Select[Range[2, 200], Mod[(# - 1)! + 1, #] == 0 &]]
46
```

Exercise 3.6

```
Select[Prime[Range[200]], Mod[19^(# - 1), #^2] == 1 &]
{3, 7, 13, 43, 137}
```

Exercise 3.7

```
re[n_] := FromDigits[Reverse[IntegerDigits[n]]]

re[12345]
54321
```

We need to look up to the 670-th prime number as

```
Prime[670]
5003

pless5000 = Prime[Range[670]];

Short[Select[pless5000, PrimeQ[re[#]] &]]
{2,3,5,7,11,13,17,31, <<151>>, 3803,3821,3851,
3889,3911,3917,3929}

Length[Select[pless5000, PrimeQ[re[#]] &]]
167
```

Exercise 3.8

```
Select[Range[1000], IntegerQ[Sqrt[#! + (# + 1)!]] &]
{4}
```

Yet another way to show there is only one n with the given property is by using pattern matching of Chapter 9.

```
Count[Sqrt[Range[1000]! + (Range[1000] + 1)!], _Integer]
1

Cases[Sqrt[Range[1000]! + (Range[1000] + 1)!], _Integer]
{12}
```

Exercise 3.9

```
Select[Range[10000], PrimeQ[#^6 + 1091] &, 5]
{3906, 4620, 5166, 5376, 5460}
```

Exercise 3.10

```
sr[n_] := Sort[IntegerDigits[n]]

cyclic[n_] := Length[Union[sr /@ (n Range[6])]] == 1

Select[Range[100000, 999999], cyclic]
{142857}
```

Exercise 4.1

```
Select[Range[10, 99], Divisible[#, Plus @@ IntegerDigits[#]] &]
{10, 12, 18, 20, 21, 24, 27, 30, 36, 40, 42, 45, 48, 50, 54, 60,
63, 70, 72, 80, 81, 84, 90}

Length[Select[Range[10000, 99999],
  Divisible[#, Plus @@ IntegerDigits[#]] &]]
10334
```

Exercise 4.3

```
FactorInteger[2^4* 5^3* 7^3]
{{2, 4}, {5, 3}, {7, 3}}

Times @@@ FactorInteger[2^4* 5^3* 7^3]
{8, 15, 21}
```

As one can guess from the code `@@@` goes into the second level of the lists and replaces the heads in that level.

```
Plus @@ (Times @@@ FactorInteger[2^4* 5^3* 7^3])
44

Select[Range[100], Plus @@ (Times @@@ FactorInteger[#]) == # &]
{1, 2, 3, 4, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,
59, 61, 67, 71, 73, 79, 83, 89, 97}
```

Exercise 4.4

```
f[n_] := 3 n^2 + n + 1

sd[n_] := Plus @@ IntegerDigits[n]

Min[sd /@ (f /@ Range[10000])]
3
```

Exercise 4.5

```
Power @@ (x + y)
x^y

Plus @@@ (x^y + y^z)
x + 2 y + z
```

Exercise 4.6 This exercise again demonstrates that *Mathematica* is quite at ease with working with large numbers. First we get all the divisors of this large number. Then we add them together. Then we need to show that this number is a perfect number.

```
t = Divisors[6086555670238378989670371734243169622657830773351885970528324860512791691264];

t1 = Plus @@ t
1447401115466452442794637312608598848157367749147483588906635434913119915212 8
```

We have seen the following code in Problem 4.5, where we discussed perfect numbers.

```
Plus @@ Most[Divisors[t1]]
1447401115466452442794637312608598848157367749147483588906635434913119915212 8
```

Exercise 6.1

```
Clear[n, k]

Sum[k/(k^4 + k^2 + 1), {k, 1, n}]
(n + n^2)/(2 (1 + n + n^2))
```

Exercise 6.2

```
Clear[f, g, n]

f[k_] := Sum[(-1)^n t^n/n!, {n, 1, k}]

g[k_] := Sum[(-1)^n n!/t^n, {n, 1, k}]

2 - f[2] g[2] == 2/t + t/2
True
```

Exercise 6.3

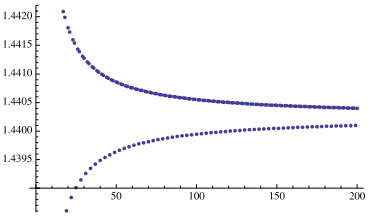
```
f[k_] := Sin[x] + Sum[x^i/i!, {i, 1, k, 2}]

f[3]
x + x^3/6 + Sin[x]
```

Exercise 6.4

```
NSum[(-1)^n/(2 n + 1) Sum[1/(2 n + 4 k + 3), {k, 0, 2 n}],
{n, 0, Infinity}]
0.250902

N[(3 Pi/8 ) Log[(Sqrt[5] + 1)/2] - Pi/16 Log[5]]
0.250902
```

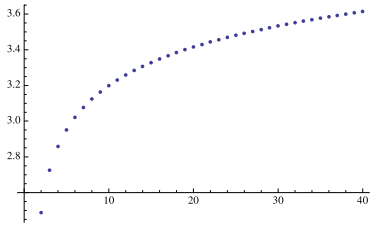
Exercise 6.5

```
t = Table[
   Sum[(-1)^i (1/(i + 1) + 1/(i + 2) + 1/(i + 3)), {i, 0, n,3}],
   {n, 10, 10000, 50}];

ListPlot[%]
```



Exercise 6.6

```
Sum[Sum[Prime[j], {j, i, 2 i - 1}]^(-1/2), {i, 1, 10}]
3/(2 Sqrt[2]) + (5 Sqrt[3])/44 + 1/(6 Sqrt[13]) + 1/Sqrt[23] +
1/( 3 Sqrt[30]) + 1/(2 Sqrt[33]) + 1/Sqrt[83] + 1/Sqrt[197]

ListPlot[Table[
  Sum[Sum[Prime[j], {j, i, 2 i - 1}]^(-1/2), {i, 1, n}],
  {n, 1, 2000,  50}]]
```



Exercise 6.7

```
Clear[f]

f[n_] := Product[Fibonacci[i] + x^i, {i, 1, n}]

f[3]
(1 + x) (1 + x^2) (2 + x^3)

Coefficient[f[23], x^4]
448751229405488344922780319810583408511498496000000
```

Exercise 7.2

```
n = t = 99999; i = 1;
While[MemberQ[IntegerDigits[t], 9],
 t = n* i++]; Print[i, "   ", t]

11113   1111188888
```

Exercise 7.3

```
re[n_] := FromDigits[Reverse[IntegerDigits[n]]]

re[12345]
54321

Do[
 Do[
  If[IntegerQ[Sqrt[m^2 + n^2]] && m == re[n], Print[n, "  ", m ]],
  {m, n, 1000}],
 {n, 1, 1000}]
```

As we don't get an answer, there is no Pythagorean pair smaller than 1000 which are reverses of each other.

Exercise 7.4

```
Input["Enter a prime number", p];
Do[
 Do[
  If[r^2 - q^2 == p^2, Print[r, "  ", q ]],
  {r, q + 1, 101}],
 {q, 1, 100}]

13  12
```

Exercise 7.5

```
f[x_] := 1/(1 + x)

Nest[f, x, 3]
1/(1 + 1/(1 + 1/(1 + x)))

NSolve[x == Nest[f, x, 10], x]
{{x -> -1.61803}, {x -> 0.618034}}
```

Exercise 7.6

```
f[n_] := Nest[Sqrt[2 + #] &, Sqrt[2], n]

f[3]
Sqrt[2 + Sqrt[2 + Sqrt[2 + Sqrt[2]]]]

f[4]
Sqrt[2 + Sqrt[2 + Sqrt[2 + Sqrt[2 + Sqrt[2]]]]]

t = N[Table[Product[f[n]/2, {n, 0, k}], {k, 1, 15}]]

{0.653281, 0.640729, 0.637644, 0.636876, 0.636684, 0.636636,
0.636624, 0.636621, 0.63662, 0.63662, 0.63662, 0.63662, 0.63662,
0.63662, 0.63662}

2./Pi
0.63662
```

Exercise 8.1

```
(a + b)^n /. (x_ + y_)^z_ -> (x^z + y^z)
a^n + b^n
```

In general, $(a+b)^n$ is not a^n+b^n. However, in Ring theory, if the characteristic of a commutative ring is n, then $(a+b)^n = a^n + b^n$

Exercise 9.3

```
Select[DictionaryLookup[],
 MatchQ[Characters[#], {___, "r", "a", "t"}] &]

{"Ararat", "aristocrat", "autocrat", "baccarat", "brat",
"bureaucrat", "carat", "democrat", "Democrat", "Dixiecrat",
"drat", "frat", "Gujarat", "karat", "Marat", "Montserrat",
"Murat", "muskrat", "plutocrat", "prat", "rat", "Seurat", "sprat",
"Surat", "technocrat"}
```

Exercise 11.1

```
a[1] = 7;
a[n_] := a[n] = a[n - 1] + GCD[n, a[n - 1]]

Do[
 If[(t = a[n] - a[n - 1]) != 1, Print[n, " gives the prime ",
 t]], {n, 2, 500}]
```

```
5 gives the prime 5
6 gives the prime 3
11 gives the prime 11
12 gives the prime 3
23 gives the prime 23

24 gives the prime 3
47 gives the prime 47
48 gives the prime 3
50 gives the prime 5
51 gives the prime 3
101 gives the prime 101

102 gives the prime 3
105 gives the prime 7
110 gives the prime 11
111 gives the prime 3

117 gives the prime 13
233 gives the prime 233
234 gives the prime 3
467 gives the prime 467
```

Exercise 11.2

```
Clear[b]

b[1] := x
b[n_] := b[n] = b[n - 2] + x^n/n!

b[9]
x + x^3/6 + x^5/120 + x^7/5040 + x^9/362880
```

Exercise 12.2 A smart way to define this function is to use the command `Partition` .

```
?Partition[list,n] partitions list into non-overlapping
sublists of length n

d[n_] := Partition[Range[n^2], n]

Table[Det[d[n]], {n, 1, 10}]
{1, -2, 0, 0, 0, 0, 0, 0, 0, 0}
```

This shows, starting from $n > 2$, the determinant is always 0.

If you insist on defining this matrix using `Array`, here is one way to do so:

```
d[n_] := Array[#2 + (#1 - 1)* n &, {n, n}]
```

Exercise 12.3

```
Clear[b, x]

b[1] = b[2] = 1; b[3] = 2;

b[n_] := b[n] = x /. (Flatten[Solve[Det[({
           {b[n - 3], b[n - 3] + b[n - 2], 1},
           {b[n - 1], b[n - 1] + x, 0},
           {0, 0, 1}
          })] == 1, x]])
```

```
b[4]
3

b[6]
11

b[123]
387052969611369560489902431082134 02
```

Exercise 13.1

```
f[x_] := Sin[x] - Exp[-Sum[Cos[k x]/k, {k, 1, 30}]] +
  Exp[-Sum[Sin[k x]/k, {k, 1, 50}]]

Plot[f[x], {x, 0, 3 Pi}, PlotRange -> All, PlotPoints -> 50,
 MaxRecursion -> 2]
```

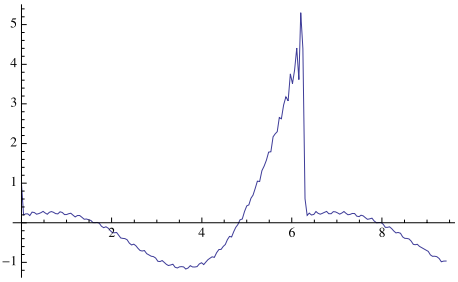


Figure 15.1 Exercise 13.1

```
NMaximize[f[x], {x, 0, 8}]
{4.47294, {x -> 6.0993}}
```

Although the maximum produced by *Mathematica* is about 4.47, a glance at the graph shows the maximum of this function is around 6. Let us draw the graph around the region where that maximum is acquired and change the interval slightly.

```
Plot[f[x], {x, 5, 6.5}, PlotRange -> All, PlotPoints -> 50,
 MaxRecursion -> 2]

NMaximize[f[x], {x, 0, 7}]
{6.07758, {x -> 6.22194}}
```

Changing the interval to $[0, 7]$ produces the correct maximum. Why is this so? Investigate this.

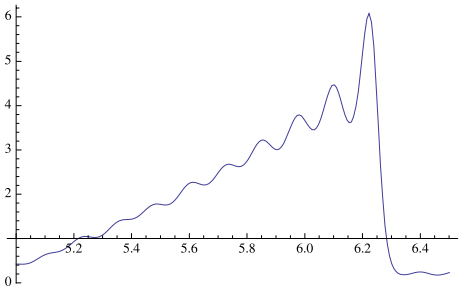


Figure 15.2 Exercise 13.1

Exercise 13.2

```
Plot[x (2 Pi - x) Sum[Sin[n x]/x, {n, 1, 50}], {x, 0, Pi},
 PlotRange -> All, PlotPoints -> 50, MaxRecursion -> 2]
```

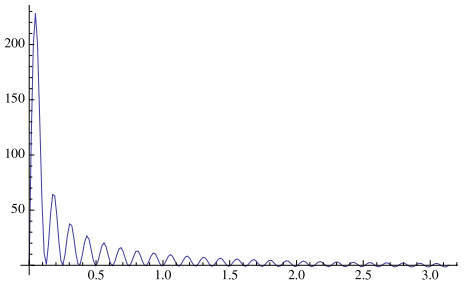


Figure 15.3 Exercise 13.2

Exercise 13.3

```
x[t_] := 4 Cos[-11 t/4] + 7 Cos[t]

y[t_] := 4 Sin[-11 t/4] + 7 Sin[t]

ParametricPlot[{x[t], y[t]}, {t, 0, 14 Pi}]
```

Exercise 13.4

```
x[t_] := Cos[t] + 1/2 Cos[7 t] + 1/3 Sin[17 t]

y[t_] := Sin[t] + 1/2 Sin[7 t] + 1/3 Cos[17 t]

ParametricPlot[{x[t], y[t]}, {t, 0, 14 Pi}]
```

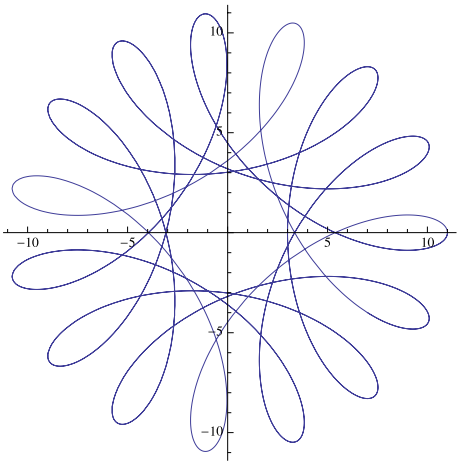


Figure 15.4 Exercise 13.3

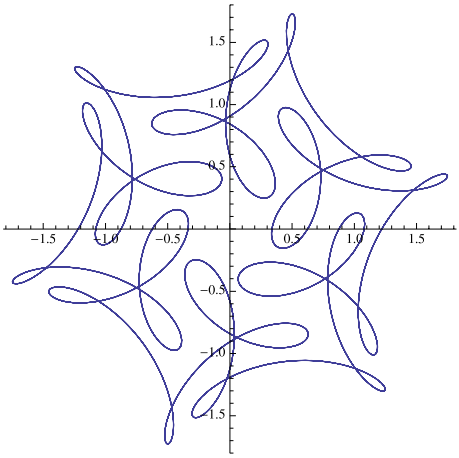


Figure 15.5 Exercise 13.4

Exercise 13.5

```
Plot[{2 Exp[-x^2], Cos[Sin[x] + Cos[x]]}, {x, -Pi, Pi}]
```

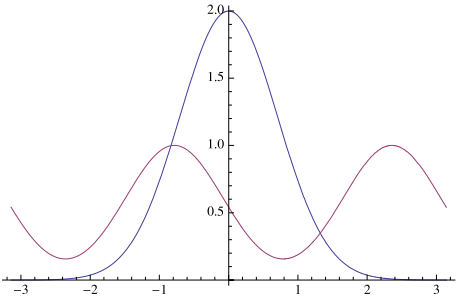


Figure 15.6 Exercise 13.5

Exercise 13.6

```
ContourPlot[ Abs[3 x^2 + x y^2 - 12] == Abs[x^2 - y^2 + 4],
{x, -10, 10}, {y, -10, 10}]
```

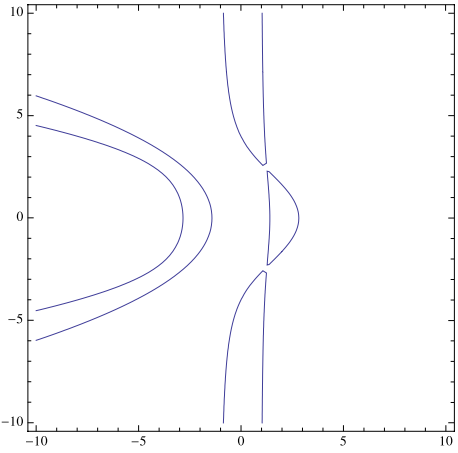


Figure 15.7 Exercise 13.6

Exercise 13.7

```
RegionPlot[Abs[x^2 + y] <= Abs[y^2 + x], {x, -5, 5}, {y, -5, 5}]
```

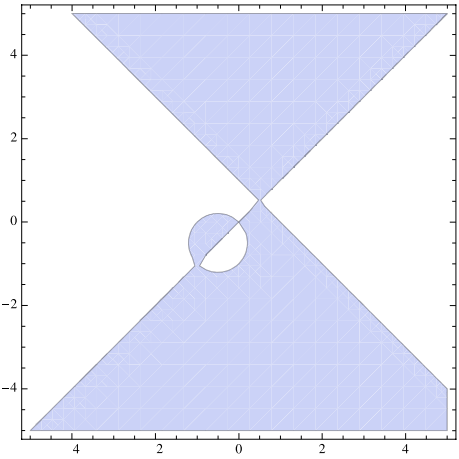


Figure 15.8 Exercise 13.7

Exercise 13.8

```
ContourPlot[4 (x^2 + y^2 - x)^3 - 27 (x^2 + y^2)^2 == 0,
{x, -4, 4}, {y, -4, 4}]
```

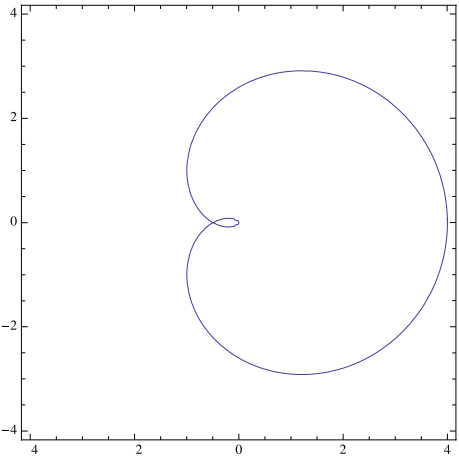


Figure 15.9 Exercise 13.8

Exercise 13.9

```
ContourPlot3D[x^2 y z + x^2 z^2 == y^3 z + y^3,
{x, -3, 3}, {y, -3, 3},  {z, -3, 3}, PlotPoints -> 50]
```

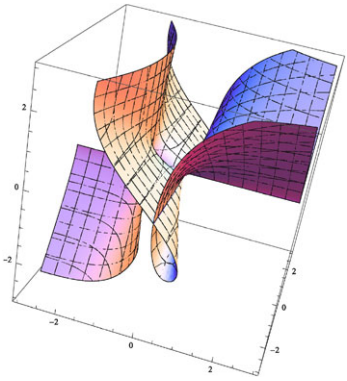


Figure 15.10 Exercise 13.9

Further reading

Wolfram *Mathematica*® provides a collection of ready to use functions, and with its rules of programming it sets the stage like a chess board. Now it depends on you (and your imagination) how to combine these and make your move to attack the problem in hand. It always helps to look at different resources to get ideas of ways to combine the *Mathematica* functions.

There are excellent books written about *Mathematica*, for example Ilan Vardi [5], Stan Wagon [6], Shaw–Tigg [4] and Gaylord, Kamin, Wellin [2] to name a few. The reader is encouraged to have a look at them.

Wolfram demonstration projects http://demonstrations.wolfram.com/ contains many interesting examples of how to use *Mathematica* in different disciplines.

And finally, the *Mathematica* Help and its virtual book is a treasure, dig it!

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3, © Springer-Verlag London Limited 2010

Bibliography

[1] R. Gaylord, *Mathematica* Programming Fundamentals, Lecture Notes, Available in MathSource 100

[2] R. Gaylord, S. Kamin, P. Wellin, An introduction to programming with *Mathematica*, Cambridge University Press, 2005. 146, 184

[3] S. Rabinowitz, Index to Mathematical problems 1980–1984, Math pro Press. 1992. viii

[4] W. Shaw, J. Tigg, Applied *Mathematica*, Addison-Wesley Publishing, 1994. 184

[5] I. Vardi, Computational Recreations in *Mathematica*, Addison-Wesley Publishing, 1991. 62, 92, 184

[6] S. Wagon, *Mathematica* in Action, Springer-Verlag, 1999. 146, 184

[7] E. Weisstein, MathWorld, `http://mathworld.wolfram.com/`. 53

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3, © Springer-Verlag London Limited 2010

Index

Abs, 62
Accumulate, 91
Algebraic, 56
Alt+., 9
And, 55
anonymous function, 23
Apart, 10
Append, 28, 59
AppendTo, 59
Apply, 48, 108
Array, 129
AxesStyle, 142

Background, 142
BarChart, 31, 44
Binomial, 8
Block, 118
Boolean expression, 54
Boolean function, 35
Booleans, 56

Cases, 102
Clear, 13
Cmd+., 9
Coefficient, 71
CoefficientList, 68
Complement, 59
Complexes, 56
ContourPlot, 135
ContourPlot3D, 153
Count, 171

defining variables, 12
Degree, 12
Delete, 28
derivation, 163
Det, 129
DictionaryLookup, 43
Divisible, 23
Divisor, 39
Divisors, 51
Do, 72
Dot, 129
Drop, 26
Dynamic, 16
dynamic variable, 15
D[], 163

Eigenvalues, 134
Eigenvectors, 134
EvenQ, 36
Exist, 57
Expand, 9

Factor, 9
FactorInteger, 6
Fibonacci, 21
Fibonacci number, 21
FillingStyle, 142
FindInstance, 158
FindRoot, 158
First, 26
FixedPointList, 87
Flatten, 27, 83
Fold, 90
FoldList, 90, 146

R. Hazrat, *Mathematica®: A Problem-Centered Approach*,
Springer Undergraduate Mathematics Series,
DOI 10.1007/978-1-84996-251-3, © Springer-Verlag London Limited 2010

For, 78
ForAll, 57
Frame, 142
FrameLabel, 142
FrameStyle, 142
FromDigits, 37
FullForm, 47
FullSimplify, 4
function, 19
– multi def's., 113
– with condition, 112

Graphics, 147
graphics, 135
GraphicsColumn, 145
GraphicsGrid, 145, 150
GraphicsRow, 145
GraphPlot, 94

Head, 48

If, 61
if statement, 61
Implies, 57
Inner, 93
inner product, 127
Input, 77
Insert, 28
IntegerDigits, 37
IntegerExponent, 111
IntegerQ, 36
Integers, 56
Integrate, 163
integration, 163
Intersection, 58
Inverse, 129

Join, 60

Last, 26
Length, 35
LengthWhile, 89
Line, 147
list, 25
listable function, 32
ListAnimate, 154
ListPlot, 149
loop, 72
– Do-loop, 72
– For-loop, 78
– nested loop, 81
– While-loop, 75

Manipulate, 17
Map, 33, 107
MatchQ, 100
matrix, 128
MatrixForm, 129
Maximize, 166
MaxIterations, 97
MaxRecursive, 145
MemberQ, 78, 105
Min, 52
Minimize, 166
Mod, 8
Module, 118
Most, 26

N, 3
NestList, 84
NestWhile, 86
NestWhileList, 86
Next, 84
NIntegrate, 163
NMaximize, 166
NMinimize, 166
Norm, 127
Not, 55
NProduct, 71
NSolve, 158
NSum, 69

OddQ, 36
Or, 55
Outer, 56, 93

palindromic, 43
ParametrixPlot, 135
ParametrixPlot3D, 153
Partition, 177
pattern matching, 100
perfect number, 51
Permutations, 98
Piecewise, 63, 143
Plot, 135
Plot3D, 153
PlotPoints, 145
PlotStyle, 142
PolarPlot, 135
Prepend, 28
Prime, 6
prime number, 2
PrimePi, 7
Primes, 56
Print, 30
Product, 70
pure function, 23

Quiet, 97
quit kernel, 9
Quotient, 9, 87

RandomInteger, 34
Range, 28
Rationals, 56
RealDigits, 146
Reals, 56
RecursionLimit, 123
Reduce, 158
RegionPlot, 135
RegionPlot3D, 153
ReplaceAll, 97
ReplaceList, 107
ReplaceRepeated, 97
Rest, 26
Reverse, 28, 37
RotateLeft, 28
RotateRight, 28
rules, 96

Select, 34, 41
Short, 45
Show, 141
Simplify, 4
Slider, 16
social number, 90
Solve, 158
solving equation, 158
Sort, 28
square free, 41
StringDrop, 45
StringLength, 45
StringReplace, 45
StringReverse, 43, 44
StringTake, 45
sublime number, 53
Sum, 65

Table, 28, 83
Take, 26
TakeWhile, 89
Tally, 60
Thread, 52
three-dimensional graph, 153
Thue-Morse seq., 98, 151
Timing, 73
Together, 10
ToString, 45
Transpose, 93
TreeForm, 49
TrigExpand, 11
TrigFactor, 11
two-dimensional graph, 135

Union, 40, 58

vector, 127

weird number, 53
Which, 61
which statement, 61
While, 75
With, 118